VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

Přímý přenos multimediálních dat mezi zařízeními s OS Android

Direct Multimedia Transfer between Android-based Mobile Devices

2015 Bc. Martin Cvečko

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Cvečko**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Přímý přenos multimediálních dat mezi zařízeními s OS Android
Direct Multimedia Transfer between Android-based Mobile Devices

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je tvorba aplikace, která bude prostřednictvím přímého propojení 2 zařízení s OS Android přenášet (pomocí WiFi sítě - HotSpot+klient, WiFi Direct, apod.) obraz, zvuk a popřípadě další informace (statické obrázky, textové poznámky, data ze senzorů) mezi 2 zařízeními. Aplikace se bude přizpůsobovat měnícím podmínkám a dynamicky adaptovat úroveň komprese, rozlišení a popř. použité kodeky a pracovat s kamerou a mikrofonem zařízení.

1. Prozkoumejte existující aplikace pro Real-Time komunikaci v prostředí WiFi sítě a možnosti, jaké nabízí.
2. Zjistěte, jaké možnosti přímé komunikace mezi 2 zařízeními a jaké možnosti pro streaming multimediálních dat platforma Android nabízí, zaměřte se na metody s co nejmenším zpožděním.
3. Analyzujte potřeby a navrhněte aplikaci, která bude snímat a přenášet multimediální data a dodatečné informace mezi 2 zařízeními s OS Android a bude se přizpůsobovat aktuálním podmínkám (konektivita, kvalita spojení, dosažitelná přenosová rychlost, atp.).
4. Aplikaci implementujte a důkladně otestujte. Vyhodnoťte zpoždění přenášených dat, rychlost zotavení z chyb a kvalitu přenášeného obrazu a zvuku.

Seznam doporučené odborné literatury:

[1] J. F. DiMarzio: Practical Android 4 Games Development. Apress, 2011, ISBN: 978-1-4302-4029-7
[2] Reto Meier: Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275
[3] Sayed Hashimi: Pro Android 2, Apress, 2010, ISBN-13: 978-1430226598
[4] Android Developers [online]. [cit. 2013-09-10]. Dostupné z: http://developer.android.com/
[5] Yadvendra, Preeti: The video streaming over Wi-Fi network application client on the Android platform. [cit. 2014-05-28] Dostupné z: http://www.ijcsits.org/papers/vol3no42013/8vol3no4.pdf

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa 6.5.2015

.....

Poďakovanie

Chcel by som poďakovať svojmu vedúcemu tejto práce, Ing. Pavlovi Marovavcovi, Ph.D., za to že ju pre mňa vôbec vypísal a pomáhal mi počas vypracovania a poskytoval mi cenné rady a odbornú pomoc.

Abstrakt

Cieľom tejto práce bolo vytvoriť aplikáciu pre operačný systém Android, ktorá by bola schopná priameho prepojenia dvoch mobilných zariadení pomocou technológie Wi-Fi a umožní prepojeným zariadeniam posielať audio a video v reálnom čase, alebo posielať statické multimediálne dáta a dáta zo senzorov. Práca popisuje existujúce metódy pre real-time streamovanie na Android platforme a zameriava sa na tie s najmenším oneskorením. Práca popisuje aj existujúce aplikácie ktoré slúžia na priame prepojenie zariadení a prenos takýchto dát. Popisuje aj základné princípy prenosu dát na Wi-Fi a aj postup ako ich implementovať na Android platforme. Nakoniec popisuje zvolený spôsob implementácie pre aplikáciu ktorá je cieľom tejto práce

Kľúčové slová:

Android, Wi-Fi, WiFi-Direct, Hotspot, stream, socket, MJPEG, HTTP

Abstract

The aim of this work is to create an application for the Android operating system capable of direct connection of two mobile devices via Wi-Fi technology, allowing connected devices transfer audio and video in real-time, or transfer static multimedia and sensor data. This work describes existing methods for real-time streaming on Android platform and it is focusing on those with the smallest delay. This work also describes existing applications which are used for direct connection of devices and transfer such data. It is also describing basic principles of data transfer on Wi-Fi and also procedures how to implement them on Android platform. Finaly this work describes chosen method of implementation for the application which is the main goal of this work.

Keywords:

Android, Wi-Fi, WiFi-Direct, Hotspot, stream, socket, MJPEG, HTTP

Zoznam použitých symbolov a skratiek

ADT	Android Developer Tools
AP	Access Point
API	Application Programming Interface
BSSID	Basic Service Set Identifier
DNS	Domain Name System
FPS	Frames Per Second
HD	High Definition
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
MJPEG	Motion JPEG
OS	Operation System
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WWW	World Wide Web
SSID	Service Set Identificator

Obsah

1. Úvod

Mobilné zariadenia ako sú napríklad smartfóny a tablety sú v dnešnej dobe súčasťou každodenného života. S tým samozrejme prichádza potreba sťahovania, uploadovania a prezerenia multimediálnyhc súborov na týchto zariadeniach. Sledovanie filmov online počas cesty do práce, posielanie fotiek priateľom, alebo využívanie mobilných aplikácií na audio/video hovory sa stáva čoraz bežnejšou vecou. Keďže rozprávame o mobilných zariadeniach, tak väčšina komunikácie medzi zariadeniami prebieha bezdrátovo, teda pomocou Wi-Fi. Platforma Android poskytuje množstvo aplikácií ktoré dokážu komunikovať s inými zariadeniami pomocou Wi-Fi sietí. Takto je možné posielať si medzi zariadeniami rôzne typy súborov, pripájať sa na vzdialené servere a sledovať online filmy alebo dokonca živé prenosy. V dnešnej dobe sú už niektoré takéto aplikácie celosvetovo rozšírené, ako napríklad Skype. Väčšina takýchto aplikácií ale neumožňuje priame prepojenie dvoch zariadení, ale len prostredníctvom servera tretej strany. Inými slovami zariadenie sa najskôr musí pripojiť na online server a ten im potom umožňuje komunikovať s ďalšími zariadeniami. Takáto komunikácia často býva náročná a závisí hlavne od rýchlosti siete na ktorej sme pripojený. V súčasnosti ešte stále existuje veľa domácností ktorých Wi-Fi siete sú nedostatočné pre kvalitné videohovory alebo prenos väčších súborov. Vo výsledku tak môže dôjsť k tomu že uskutočnený videohovor bude mať zlú kvalitu obraz, oneskorený prenos zvuku, pomalý prenos dát. Problémom dnešnej doby je to že Wi-Fi siete pokrývajú len niektoré oblasti. Problém môže nastať napríklad v situácii kedy potrebujeme prepojiť 2 zariadenia ktoré sú síce pomerne blízko seba ale nie je dostupná žiadna Wi-Fi sieť. Ak napríklad robotník stojí na stožiari a premeriava signál alebo niečo opravuje a potrebuje ukázať nejaký detail kolegovi čo stojí na zemi pod ním potrebujú aplikáciu ktorá prepojí ich zariadenia priamo bez internetového pripojenia. Ďalším príkladom by mohlo byť vytvorenie kamery. Nepotrebné mobilné zariadenie môže slúžiť ako kamera na ktorú sa dá pripojiť či už s internetovým pripojením, alebo bez neho. Využitie môže byť ale aj úplne triviálne, ako napríklad komunikácia dvoch ľudí na rôznych poschodiach budovy. Keďže nevyužívame prepojenie dvoch zariadení cez tretí server ale prepájame ich priamo, prenosová rýchlosť bude vo väčšine prípadov oveľa lepšia. V prípadoch kedy potrebujeme vytvoriť komunikáciu na kratšiu vzdialenosť je teda rozumnejšie využívať aplikácie ktoré umožňujú priame prepojenie zariadení.

Cieľom tejto práce je preskúmať možnosti priameho prepojenia dvoch zariadení s operačným systémom Android pomocou Wi-Fi a navrhnúť a implementovať aplikáciu ktorá bude schopná prenášať audio a video v reálnom čase a prípadne aj iné statické multimediálne dáta ako obrázky, videá, audio súbory, alebo dáta zo senzorov. Jedným z kritérií implementovanej aplikácie je zameranie sa na metódy prenosu dát s čo najmenším oneskorením.

V úvodnej kapitole sa oboznámime s pojmom multimédia a čo to znamená streamovanie. Dozvieme sa aké druhy multimédií a streamov existujú a aké majú v dnešnej dobe využitie. Takisto si

priblížime niekoľko aplikácií ktoré využívajú multimédia a streamy a majú podobnú funkčnosť ako cieľová aplikácia tejto práce. Ďalšia kapitola je zameraná na základy ohľadne komunikácie pomocou technológie Wi-Fi. V tejto kapitole sú popísané základy komunikácie a posielania dát založené na Wi-Fi. V rámci tejto kapitoly sú popísané aj rôzne typy Wi-Fi komunikácie. Štvrtá kapitola sa venuje možnostiam real-time streamovaniu na platforme Android a ich základnému popisu. Piata kapitola sa už venuje vývoji samotnej aplikácie ktorá je cieľom tejto práce. Kapitola popisuje postup pri vývoji aplikácie. Zdôvodňuje niektoré rozhodnutia pri výbere metód použitých na docielenie požadovanej funkčnosti a popisuje aj postupy pri vytváraní kľúčových funkcií aplikácie. Šiesta kapitola sa venuje testovaniu cieľovej aplikácie. Obsahuje výsledky vykonaných testov aplikácie za rôznych okolností ktoré sú prezentované formou grafov. Posledná kapitola je záver kde je aplikácia zhodnotená ako celok. Hodnotí sa či odpovedá predpokladom a požiadavkám ktoré sme mali a prípadne nedostatky aplikácie alebo niektoré funkcie ktoré by sa v aplikácii ešte do budúcnosti mali dopracovať.

1 Multimédia a streamovanie

Pojem multimédia popisuje obsah, ktorý je vytvorený z kombinácie rôznych foriem obsahu. Multimediálne zahŕňajú kombináciu zvuku, textu, videa, animácií, statických snímok, alebo interaktívnej formy obsahu. Multimédia sú bežne nahrávané, prehrávané, alebo sprístupnené informačným zariadením ktoré spracováva obsah, ako sú napríklad počítačové alebo elektronické zariadení, ale môžu byť aj súčasťou živého vystúpenia.[18]

Pojem multimediálne aplikácie alebo multimediálny softvér, sa začalo používať od začiatku 90. rokov 20. storočia, a označoval kombinácie textových, obrazových, zvukových alebo animovaných alebo filmových dát. Konzorcium pod vedením spoločnosti Microsoft vydalo v roku 1991 špecifikáciu štandardného multimediálneho počítača (MPC). Tá bola v ďalších rokoch niekoľkokrát aktualizovaná, dnes sú ale prakticky všetky osobné počítače multimediálne.

Multimediálny kontajner je označenie pre dátový tok alebo obálku súboru, obsahujúci jeden alebo viac prúdov multimediálnych dát (streamov). Do jedného súboru je teda možné napríklad uložiť jednu video stopu, niekoľko zvukových stôp v rôznych jazykoch a niekoľko titulkov a je zaistená ich synchronizácia. Užívateľ si tak pri prehrávaní môže vybrať, ktorú kombináciu multimediálnych dát chce použiť.[18]

Multimédia nachádzajú uplatnenie v rôznych oblastiach, ako sú napríklad reklamy, umenie, vzdelávanie, zábava, strojárstvo, medicína, matematika, obchod, vedecký výskum a mnohé ďalšie oblasti.

1.1 Streamovanie

Streaming (z anglického stream - prúd) je označenie pre technológiu kontinuálneho prenosu audiovizuálneho materiálu medzi zdrojom a koncovým užívateľom. V súčasnosti sa streaming používa hlavne na prenášanie audiovizuálneho materiálu na internete (webcasting). Webcasting môže fungovať v reálnom čase (internetová televízia alebo rádio), alebo na princípe Video on demand (napr. YouTube). Pre streamovanie videa viacerým používateľom zároveň musí mať prevádzkovateľ k dispozícii okrem obsahu aj server pre streamovanie, ktorý zabezpečuje komunikáciu s cieľovými zariadeniami a plynulé vysielanie dát.[19]

Streamovacie médium je multimédium ktoré je neustále prijímané a predkladané ku koncovému užívateľovi, zatiaľ čo je dodávané poskytovateľom. Klientov prehrávač médií môže začať prehrávať dáta (napr. film), akoby bol prenesený celý súbor. Rozlišovanie spôsobu doručenia z distribuovaných médií sa vzťahuje špecificky k telekomunikačným sieťam, ako väčšina ostatných aplikačných systémov

sú buď čisto streamovacie (napr. rádio, televízia) alebo nestreamovacie (napr. knihy, videokazety, audio CD). Termín streaming media možno použiť aj na iné média ako je video a audio, ako sú napr. živé skryté titulky, burzové pásky, a real-time texty, ktoré sú považované za "streaming text". Termín streaming bol prvýkrát použitý v roku 1990 ako popis pre video na vyžiadanie na IP sieťach. V tomto čase bolo takéto video zvyčajne označované ako store and forward video čo bolo zavádzajúca názvoslovie. [18]

Kvalita videa

Na prenos audiovizuálnych dát po internete sú nutné kodeky ktoré zmenšujú objem dát. Streamovanie najviac využíva flashových kodekov, MPEG-4, Windows Media, Real-Time a Quick Time. Aj tak je prenos záznamu v televíznom rozlíšení (720 × 576) veľmi náročný. Preto bolo pre streamovanie najpoužívanejšie rozlíšenie 320 × 240 bodov pri dátovom toku 256-512 Kbps. V súčasnosti prenos vo vysokom rozlíšení poskytujú viaceré servere ako napríklad Youtube, MySpace, Stream.cz atd. [10]

Kvalita audia

Pre streamovanie audia sa používajú hlavne kodeky MP3, Windows Media Audio (WMA), AAC+, OGG, ktoré sú v dátových tokoch od 16-256 kbps. Audio môže byť streamované ako single bitrate, teda jeden konštantný dátový tok alebo multibitrate, teda viac konštantných dátových tokov prenášaných spoločne v jednom dátovom toku medzi kodérom streamu a serverom. Prehrávač podporujúce multibitrate stream zo servera dokážu automaticky meniť kvalitu zvuku v prípade zhoršenia / zlepšenie kvality internetového pripojenia.[10]

Pseudostreaming

Pretože streamovanie je zo strany prevádzkovateľa streamovacieho servera finančne aj technologicky náročné, bola vyvinutá jednoduchšia alternatíva pre streamovanie audiovizuálneho záznamu, ktorá sa nazýva pseudostreaming. Pseudostreaming simuluje správanie skutočného streamingu, a to najmä na strane servera. Svojimi možnosťami a schopnosťami sa zďaleka nevyrovná skutočnému streamingu. Cieľom použitia pseudostreamingu je zvyčajne dosiahnuť s prijateľnými nákladmi a štandardnými softvérovými technológiami, schopnosť v prehrávači záznamu na strane klienta, prechádzať v časovej osi audiovizuálneho záznamu na akúkoľvek pozíciu bez toho, že by užívateľ musel čakať na načítanie dát celého záznamu.[18]

Kodek

Kodek (zloženina z počiatočných slabík slov "kodér a dekodér", respektíve kompresia a dekompresia) je zariadenie alebo počítačový program, schopný transformovať stream alebo signál.

Kodeky ukladajú dáta do zakódovanej formy (hlavne kvôli prenosu, uchovávaniu alebo šifrovania), ale väčšinou sa používajú pre obnovenie presnej alebo približnej pôvodnej formy dát vhodnej pre zobrazovanie, alebo manipuláciu. Kodeky sú základnou súčasťou softvéru pre editáciu multimediálnych súborov ako je hudba alebo filmy a často sa používajú pre videokonferencie a distribúciu multimediálnych dát v sieťach teda streamovanie.

Sieťovo šírená multimédiá väčšinou obsahujú niekoľko častí. Zvukové aj obrazové dáta a k ním dodatočné informácie (metadáta), podľa ktorých je možné obe zložky synchronizovať. Každá z častí môže byť určená pre iný program, alebo proces. Aby bolo možné s nimi manipulovať, musia byť zapuzdrené v spoločnom celku.

Súčasťou šírených dát môže byť aj obálka, ktorá sa na rozdiel od metadát nepodieľa na informačnom obsahu. Pridáva sa kvôli sprístupneniu informácií alebo aj kvôli väčšej robustnosti dátového toku. Aby sa samotné zakódované zvukové a obrazové dáta odlíšili od ostatných súčastí dátového toku, často sa nazýva esencia.

Kodeky sú často navrhované tak, aby zdôraznili niektoré aspekty médií, ktoré majú kódovať. Napríklad digitálne video (kodek DV) športových udalostí musí kódovať kvalitne pohyby, ale nie nevyhnutne presné farby, naopak video z umeleckej výstavy musí kódovať čo najkvalitnejšie farby a štruktúry povrchu.

Audio kodeky pre mobilné zariadenia musia mať veľmi nízku latenciu medzi kódovaním zdroja a prehrávaním. Naproti tomu možno audio kodeky pre záznam alebo vysielanie pomocou vysokej latenciu kompresnej techniky audia dosahujú vyššiu presnosť pri nižšom bit-rate.

Mnohé multimediálne dátové streamy obsahujú aj audio aj video a často nejaké metadáta umožňujúce synchronizáciu zvuku a videa. Každý z týchto troch streamov môže byť riešený prostredníctvom rôznych programov, procesov, alebo hardvéru. Ale aby boli multimediálne dátové prúdy užitočné v archivovanej alebo prenášanej forme, musia byť zapuzdrené spoločne vo formáte kontajnera.

Dopyt po streamoch

Pokroky v počítačových sieťach, v kombinácii s výkonnými domácimi počítačmi a modernými operačnými systémami, spravili streamovacie média praktické a cenovo dostupné pre bežných spotrebiteľov. Samostatné rádiové zariadenia s podporou internetu ponúkajú ich užívateľom možnosť počúvať audio streamy bez nutnosti mať počítač. Tieto audio streamovacie služby sa v posledných rokoch stávajú viac a viac populárnymi a streamovanie hudby dosiahlo rekordu až 118.1 bilióna audio streamov. Všeobecne platí, že multimediálny obsah má veľký objem, a s tým rastú náklady pre

ukladanie a prenos. Malou kompenzáciou je to že média sú zvyčajne komprimované aj pre ukladanie aj pre streamovanie.

Rastúci dopyt spotrebiteľov pre streamovanie vo vysokom rozlíšení (HD) viedol k rozvoji rady technológií, ako je WirelessHD alebo ITU-T G.hn, ktoré sú optimalizované pre streamovanie HD obsahu bez nutnosti užívateľa používať sieťové káble.

Dnes je možné média streamy prenášať buď naživo (Live stream) alebo na vyžiadanie (on-demand). Live streamy sú vo všeobecnosti poskytované prostriedkami s názvom "true streaming". True streaming odošle informácie priamo do počítača alebo zariadenia bez uloženia súboru na pevný disk. On-demand streaming je vybavený prostriedkom s názvom progressive streaming (progresívne streamovanie) alebo progressive download (progresívne sťahovanie). Progressive streaming uloží súbor na pevný disk a potom je prehrávaný z tohto umiestnenia. On-demand streamy sú často uložené na pevné disky a servery pre rozšírenie trvania, zatiaľ čo živé streamy sú k dispozícií iba v jednom okamihu iba (napr. v priebehu futbalového zápasu)

Streamovanie je čoraz viac spojené s využitím sociálnych médií. Napríklad miesta, ako je YouTube, podporujú sociálnu interakciu vo webových vysielaniach prostredníctvom funkcií, ako je live chat, on-line prieskumy atď. Okrem toho, streamovanie médií je stále viac používané pre sociálne podnikanie a e-learning.

Šírka pásma a úložisko

Unicast pripojenia vyžadujú viac pripojení z rovnakého streamovacieho serveru, dokonca aj pri streamovani rovnakého obsahu. Rýchlosť 2.5 MB za sekundu, alebo viac, je doporučená rýchlosť pre streamovanie filmov napríklad pre Google TV, Apple TV, alebo Sony TV. Pre streamovanie vo vysokom rozlíšení je doporučená rýchlosť 10 MB za sekundu. Veľkosť úložiska streamovacieho média sa vypočítava zo šírky pásma a dĺžky média s použitím nasledujúceho vzorca (pre jedného užívateľa a súboru)

veľkosť úložiska(MB) = dĺžka(sekundy) × prenosová rýchlosť(bit/s) / (8 × 1024 × 1024)

Reálne príklady:

Jedna hodina videa kódovaného na 300 kbit / s (to je typické širokopásmové video v roku 2005 a bol zvyčajne zakódovaný v 320 × veľkosť okna 240 pixelov) bude :

(3600 s × 300000 bit/s) / (8 × 1024 × 1024) vyžaduje okolo 128 MB pamäte.

Ak je súbor uložený na serveri pre on-demand streamovanie a tento prúd je zobrazený 1.000 ľudí súčasne pomocou unicast protokolu, požiadavka je:

300 kbit/s × 1,000 = 300,000 kbit/s = 300 Mbit/s šírky pásma

To zodpovedá približne 135 GB za hodinu. Použitie multicast protokolu server vysiela iba jeden dátový prúd, ktorý je spoločný pre všetkých užívateľov. Preto taký prúd bude používať iba 300 kbit/s.

Výpočet pre streamovanie v reálnom čase je podobný.
Predpoklady: rýchlosť na kóderu je 500 kbit/s
Ak prehliadka trvá po dobu 3 hodín s 3000 divákmi, potom výpočet je:

Počet MBs prenesených = rýchlosť snímača (v bit/s) × počet sekúnd × počtom divákov / (8 * 1024 * 1024)

Počet MB prenesených = 500 x 1024 (bit/s) × 3 × 3600 (=3 hodiny) × 3000 (diváci) / (8*1024*1024) = 1977539 MB

[18]

1.2 Protokoly využívané pri streamovani

Streamovanie zahŕňa protokoly na niekoľkých rôznych vrstvách ISO-OSI modelu. Každý protokol má svoje výhody aj nevýhody a pri streamovaní dát je nutné zvážiť ktorý protokol je vhodné použiť na danný typ prenosu.

Transmission Control Protocol

TCP je spojovo orientovaný, spoľahlivý komunikačný protokol transportnej vrstvy. Tvorí prechodovú vrstvu medzi protokolom IP pod ním a aplikáciami nad ním. TCP funguje trojfázovo. Nadviazanie spojenia, poslanie dát a nakoniec kontrola dát. TCP vykonáva kontrolu posielaných paketov aby sa uistil že sa žiaden nestratí. Každému paketu priradí poradové číslo ktoré prijímané zariadenie skontroluje a pošle späť potvrdenie že dáta boli doručené. U protokolu TCP teda ide skôr o presnosť ako čas a tak často vzniká veľké oneskorenie, radovo v sekundách.

TCP prijíma data zo streamu, rozdeľuje ich na kúsky (chunks) a pridáva im TCP hlavičky a vytvára tak TCP segment. TCP segment je potom zapuzdrený do datagramu Internetového Protokolu.

TCP segment pozostáva z hlavičky segmentu a dátovej sekcie. Hlavička sa skladá z 10 povinných polí a voliteľného rozširujúceho poľa. Dátová časť nadväzuje na hlavičku. Obsahuje užitočné dáta prenášané pre aplikáciu. [16]

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Obrázok č. 1: Hlavička TCP [16]

Protokol TCP by sa mal využívať v prípadoch keď nám záleží na tom aby neboli dáta počas komunikácie stratené, alebo v zlom poradí. TCP sa využíva vo veľkej miere v mnohých populárnych aplikáciách na internete, vrátane World Wide Web (WWW), E-mail, File Transfer Protocol, Secure Shell, zdieľanie súborov peer-to-peer a mnoho ďalších.

User Datagram Protocol

UDP, je tzv. "nespoľahlivý" protokol. Takisto ako TCP patrí do transportnej vrstvy ISO-OSI modelu. Slúži na posielanie datagramov na zariadenia v sieti. Od protokolu TCP sa líši tým že nezaručuje, že posielaný paket sa po ceste nestratí, alebo sa nezmení poradie paketov, alebo že sa daný paket nedoručí viackrát. Pretože na rozdiel od TCP protokolu tieto kontroly nemá, často sa využíva v prípadoch kedy takáto kontrola nie je nevyhnutná a naopak záleží na rýchlosti prenosu. Pomocou UDP môžu aplikácie posielať správy(datagramy) na protokole IP ostatným sieťovým zariadeniam bez nutnosti vytvorenia prenosového kanálu alebo dátovej cesty.

Aplikácie používajú datagramové sockety na vytvorenie host-to-host komunikácie. Aplikácia spojí socket s koncovým bodom dátového prenosu, čo je kombinácia IP adresy a portu.[16].

Hlavička UDP pozostáva zo 4 polí, pričom každé má veľkosť 2 byty (16 bitov).

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port														
4	32	Length																Checksum														

Obrázok č. 2: Hlavička UDP [16]

UDP je vhodný pre účely keď kontrola chýb a oprava buď nie je nutná alebo sa vykonáva na úrovni aplikácie aby sa predišlo preťaženiu siete. Aplikácie v ktorých je prioritou skôr čas ako kvalita často využívajú UDP pretože je lepšie packet stratiť ako čakať na oneskorený paket. UDP sa používa napríklad na DNS, streamované médiá, prenos hlasu alebo videa (VoIP) a online hry.

Real-time Transport Protocol

RTP je protokol ktorý na rozdiel od TCP a UDP patrí do aplikačnej vrstvy ISO-OSI modelu. RTP je transportný protokol ktorý zabezpečuje prenos dát, hlavne audia a videa, v reálnom čase. Presnejšie zaisťuje doručenie segmentov v správnom poradí pomocou časových razítok (timestampov), sekvenčných čísel a podobne. Protokol poskytuje služby na identifikáciu obsahu, sekvenčné číslovanie, časové pečiatky a monitorovanie doručenia. Je treba povedať, že RTP sám o sebe neposkytuje žiadny mechanizmus ktorý zaisťuje včasné doručenie dát alebo poskytovania QoS (Quality of Service), ale spolieha sa na služby nižších úrovní. Typicky beží RTP na protokole UDP, môže byť ale použitý aj v iných sieťových a transportných protokoloch. Sekvenčné číslo obsiahnuté v RTP umožňuje prijímateľovi rekonštruovať postupnosť segmentov od odosielateľa, môže byť ale tiež použité na určenie správnej pozície segmentu, napr. pri dekódovaní videa bez nutnosti dekódovať predchádzajúce segmenty. Pri prenose informácií v reálnom čase je možné zanedbať určitú stratu paketov, prípadne ich doručenie v nesprávnom poradí, ak takto dosiahneme menšie oneskorenie pri samotnom prenose dát.[16]

RTP hlavička ma minimálnu veľkosť 12 bytov. Po hlavičke môže nasledovať voliteľná rozšírená hlavička. Potom nasleduje užitočný obsah RTP paketu, formát ktorý je určený v závislosti na konkrétnej triede aplikácie.

Bit offset[b]	0-1	2	3	4-7	8	9-15	16-31
0	Version	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	CSRC identifiers ...						
96+32×CC	Profile-specific extension header ID						Extension header length
128+32×CC	Extension header ...						

Obrázok č. 3: Hlavička RTP[16]

Real-time Streaming Protocol

RTSP podobne ako RTP patrí medzi protokoly patriace do aplikačnej vrstvy ISO-OSI modelu. RTSP je sieťový protokol navrhnutý pre použitie v rôznych zábavných a komunikačných systémoch pre riadenie serverov streamujúcich média. Protokol sa využíva na vytvorenie a riadenie relácie (session) medzi koncovými bodmi. Prenos streamovaných dát sám o sebe nie je úlohou RTSP protokolu. I keď v niektorých smeroch podobný protokolu HTTP, RTSP definuje kontrolné sekvencie ktoré majú využitie pri prehrávaní multimédií. Podobne ako HTTP, RTSP využíva TCP na udržiavanie end-to-end spojenia a zatiaľ čo väčšina kontrolných správ je posielaná klientom na server, niektoré príkazy sú posielané v opačnom smere. Väčšina RTSP serverov využíva RTP protokol v spojení s real-time control protocol (RTCP) pre dodávania streamovaného média, avšak niektorý predajcovia implementujú vlastné prenosové protokoly. [16]

Hypertext Transfer Protocol

Ďalším protokolom aplikačnej vrstvy modelu ISO-OSI je HTTP. HTTP je protokol definujúci požiadavky a odpovede medzi klientmi a servermi. Klient, čo môže byť napríklad webový prehliadač, väčšinou začne požiadaním o nadviazanie spojenia na určenom porte vzdialeného stroja (bežne sa používa port 80). Server ktorý na danom porte počúva, čaká, kým klient pošle požiadavok ako "GET / HTTP/1.1" (ktorý žiada o zaslanie štartovacej stránky webservera) nasledovaný sériou hlavičiek podobných MIME opisujúcich detaily požiadavku a za nimi objektom zvolených údajov.

Typ internetového média, pôvodne označovaný ako typ MIME je štandardný internetový identifikátor určujúci typ dát obsiahnutých v súbore. Vo webových prehliadačoch zabezpečujú aby zobrazovali a spracovávali súbory ktoré nie sú vo formáte HTML[16]

Po prijatí požiadavky server pošle reťazec s odpoveďou ako "200 OK" nasledovanou hlavičkami spolu so samotnou správou, ktorej telo tvorí obsah požadovaného súboru, chybové hlásenie alebo iná informácia.

Ukážka jednoduchého klientskeho požiadavku:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Požiadavka klienta je nasledovaná prázdnym riadkom, takže požiadavka končí s dvojitým znakom pre nový riadok. Políčko HOST rozlišuje medzi rôznymi DNS názvami ktoré zdieľajú jednu IP adresu. Zatiaľ čo v HTTP 1.0 bola táto hlavička nepovinná, HTTP 1.1 ju už vyžaduje.[16]

Odpoveď servera na ukážkový request:

```
HTTP/1.1 200 OK
```

```
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Accept-Ranges: bytes
Connection: close

<html>
        <head>
                <title>An Example Page</title>
        </head>
        <body>
                Hello World, this is a very simple HTML document.
        </body>
</html>
```

Čo sa týka streamovania dát, HTTP protocol je využívaný pri jednom zo spôsobov real-time komunikácie a tým je MJPEG posielaný cez HTTP protokol.

1.3 Aplikácie pre prenos multimediálnych dát

Whatsapp

WhatsApp Messenger je mobilná aplikácia ktorá slúži na posielanie správ. Je dostupná pre operačné systémy ako Android, iOS, Windows Phone, BlackBerry a ďalšie. WhatsApp používa pre odosielanie správ siete 3G alebo Wi-Fi (ak sú dostupné). Obsahom správy môžu samozrejme byť aj obrázky a audio alebo video súbory. WhatsApp používa upravenú verziu otvoreného štandardu Extensible Messaging and Presence Protocol (XMPP). Pri inštalácii vytvorí užívateľský účet pomocou telefónneho číslo ako užívateľského mena (Jabber ID:.[Telefónne číslo]@s.whatsapp.net). Softvér WhatsApp automaticky porovnáva všetky telefónne čísla z adresára na zariadení s centrálnou databázou užívateľov WhatsApp a automaticky pridá kontakty do WhatsApp zoznamu kontaktov užívateľa. Skoršia verzia Androidu používala MD5-hash, reverznú verziu telefónneho IMEI ako heslo, zatiaľ čo verzia iOS používala v telefóne Wi-Fi MAC adresu namiesto IMEI. Aktualizácia v 2012 teraz vygeneruje náhodné heslo na strane servera

WhatsApp je podporovaný na väčšine Android, BlackBerry, iPhone a Nokia smartphonov. Všetky telefóny Android bežiace na Android 2.1 a vyššie, všetky zariadenia BlackBerry s operačným systémom OS 4.7 a novšie, vrátane BlackBerry 10 a všetky iPhony so systémom iOS 4.3 a novšie. Avšak, niektoré zariadenia Dual SIM nemusia byť kompatibilné s WhatsApp.

Multimediálne správy sú odosielané nahrávaním obrazu, zvuku alebo videa ktoré bude odoslané na HTTP server a potom odošle odkaz na obsah spolu s jeho base64 zakódovanou miniatúrou (ak je k dispozícii).

Výhody WhatsApp:

- Akonáhle si stiahnete aplikáciu, môžete posielať správy bez obmedzenia – ale len prvý rok, potom už je aplikácia platená
- Odoslanie videa, obrázkov a hlasových záznamov svojím kontaktom.
- Umožňuje vytvárať skupinové rozhovory s vašimi kontaktmi .
- Autorizácia: WhatsApp pracuje s telefónnym číslom, rovnako ako SMS sa bezchybne prepojí s vaším existujúcim telefónnym adresárom. Nie je nutné žiadne prihlasovacie meno ani heslo.
- Prihlasovanie: žiadny zmätok ohľadne prihlasovania a odhlasovania z iného počítača alebo zariadenia. WhatsApp používa push notifikácie a je stále pripojený.
- Zoznam kontaktov v aplikácii je automaticky prepojený s kontaktmi telefónu – ale len tie kontakty ktoré používajú WhatsApp. Nie je potreba manuálne vyhľadávanie.
- Offline správy: Aj keď ujde push oznámenie alebo sa vypne telefón, WhatsApp bude ukladať svoje správy v režime offline, kým ich znovu nezískate pri ďalšom použití aplikácie.
- Ďalšie výhody: Výmena kontaktov, vlastné tapeta, vlastné zvukové upozornenia, e - mail história konverzácie, ...

[17]

Viber

Viber je cross-platformová aplikácia pre posielanie správ ktorá pracuje s protokolom Voice-over-Internet Protocol (VoIP) vyvinutá spoločnosťou Viber Media. Okrem textových správ, si môžu používatelia vymieňať obrázky, video a audio správy. Klientsky softvér je k dispozícii pre Mac OS, Android, BlackBerry OS, iOS, Series 40, Symbian, Bada, Windows Phone a Microsoft Windows. Linux verzia je vo vývoji, pričom je vypustená beta verzia. Vıber funguje na oboch sieťach 3G/4G a Wi-Fi. Ako prvé vyžaduje inštaláciu na mobilnom telefóne aby sa s ňou dalo pracovať aj v prostredí desktopového operačného systému. Vıber dosiahla 7. mája 2013 cez 200 miliónov užívateľov. Viber je v súčasnosti dostupná v 30 jazykových lokalizáciách.

Skutočná funkčnosť sa mení z platformy na platformu pričom iOS a Android ako prvý dostávajú nové funkcie. Viber zahŕňa textovú, obrazovú a video komunikáciu naprieč všetkými platformami, ale hlasové hovory má k dispozícií len pre iPhone, Android a Microsoft Windows Phone. Pri inštalácii vytvorí užívateľský účet pomocou telefónneho čísla ako užívateľského mena. Viber synchronizuje s telefónnym adresárom, takže užívatelia nemusia pridávať kontakty v samostatnej knihe. V decembri

2013 Viber oficiálne zahájil Viber Out, čo je funkcia, ktorá umožňuje všetkým užívateľom volanie na mobilné a pevné linky.

V aplikácii Viber je vaše telefónne číslo vašou identifikáciou. Aplikácia synchronizuje kontakty vo vašom mobilnom zariadení a automaticky zisťuje, ktorý z vašich kontaktov používa Viber.

Výhody Viber:

- Posielanie správ medzi užívateľmi
- Hovory v kvalite zvuku HD
- Zdieľanie fotografií, videí, hlasových správ, informácií o polohe, nálepky a emotikony.
- Vytváranie skupín až 100 účastníkov
- Vďaka push oznámeniam služby nikdy nezmeškáte správu ani hovor, dokonca ani keď je aplikácia Viber vypnutá
- Podpora pre aplikáciu Viber pre počítače s operačným systémom Windows a Mac
- Aplikácia Viber je úplne zadarmo a bez reklám.

[20]

Skype

Skype je VoIP služba a instant messaging klient, v súčasnej dobe vyvinutý spoločnosťou Microsoft Skype divízie. Registrovaný užívatelia Skypu sú identifikovaný unikátnym Skype menom a možu byť uvedený v adresári Skypu. Skype mal približne 300 miliónov aktívnych užívateľov podľa štatistík z januára 2015. Táto služba umožňuje hlasovú komunikáciu pomocou mikrofónu, video hovory pomocou webovej kamery a instant messaging cez internet, ale aj tradičné telefónne hovory. Skype sa tiež stal populárny pre svoje ďalšie funkcie, vrátane prenosu súborov a videokonferencie.

Na rozdiel od väčšiny ostatných služieb na báze VoIP, Skype je hybridný systém peer-to-peer a klient-server. Využíva spracovanie na pozadí na počítačoch so softvérom Skype.

Skype používa vlastný internetový protokol s názvom Skype protocol. Protokol nebol Skypom sprístupnený verejnosti a oficiálne aplikácie používajúce tento protokol sú closed-source. Časť technológie Skype je založená na P2P protokolu Global Index, ktorý patrí k Joltid Ltd spoločnosti. Hlavný rozdiel medzi Skypom a štandardným VoIP klientom je, že Skype pracuje na modeli, peer-to-peer, a nie na obvyklom modeli klient-server (aj veľmi obľúbený model SIP VoIP je tiež peer-to-peer, ale implementácia väčšinou vyžaduje registráciu so serverom, rovnako ako Skype).

Skype podporuje len protokol IPv4. Tu chýba podpora pre internetový protokol novej generácie, IPv6.

Okrem audio kodekov G.729 a SVOPC, Skype pridal svoj vlastný audio kodek nazvaný SILK od verzie 4.0. SILK. Dodatočne Skype vydal Opus, čo je open-source kodek s integrovanými princípmi SILK-u pre prenos hlasu so zásadami CELT kodekov pre vyššiu kvalitu zvukových prenosov (napr. živé hudobné vystúpenia).

Ako video kodek je pre verzie do 5.5 používaný VP7. Od verzie 5.7 sa používa VP8 a to aj pre videá so štandardným rozlíšením aj pre 720p a 1080p high-definition videá.

Výhody Skypu:

- Telefonovanie v rámci Siete Skype
- Instant messaging – zasielanie správ a súborov medzi užívateľmi siete
- SkypeOut - platená služba pre telefonovanie do tradičných telefónnych sieti
- SkypeIn - platená služba, kedy je účastníkovi pridelené telefónne číslo, na ktoré sa možno dovolať z tradičných telefónnych sieti
- Hlasová pošta - platená služba, poskytuje funkčnosť Hlasovej schránky
- Skype Video Calling - videokonferencie medzi užívateľmi Skype, dostupné od verzie 2.0.
- Skype SMS - platená služba, umožňuje posielať SMS na mobilné telefóny
- Skype Prime - Od verzie 3.1 . Umožňuje nechať si platiť za prichádzajúce Hovory (vyžaduje ucet u PayPal)
- Presmerovanie - Umožňuje presmerovať hovory, keď nie ste on-line, je spoplatnené iba na klasické telefóny
- Skype Extras - Doplnkové Programy ako hry, nahrávanie hovoru, zdieľanie pracovnej plochy

[21]

LINE

LINE je komunikačná aplikácia, ktorá umožňuje zadarmo telefonovať a posielať správy kedykoľvek a kdekoľvek. Takisto umožňuje VoIP hovory a vytváranie audio alebo video konferencií. LINE má viac ako 450 miliónov užívateľov po celom svete a je používaný vo viac než 231 krajinách. LINE bola hodnotená ako najsťahovanejšia aplikácia v 52 krajinách. LINE bola pôvodne vyvinutá ako mobilná aplikácia pre Android a iOS smartphony. Táto služba sa od tej doby rozšírila do BlackBerry (august 2012), Nokia Asha (Ázia a Oceánia, marec 2013), Windows Phone (júl 2013) a Firefox OS (február 2014). Aplikácia existuje aj vo verziách pre notebooky a stolné počítače pre Microsoft Windows a Mac OS platformy.

Výhody LINE:

- Multiplatformná

- Voliteľná adresárová synchronizácia
- Real-time potvrdenie pri odosielaní a doručení správ
- Zdieľanie fotografií, videí a hudby s ostatnými užívateľmi
- Posielanie lokalizačných správ
- Vytvorenie a prepojenie skupín, kde užívatelia môžu chatovať a zdieľať médiá
- Nástenky (až 100 osôb naraz) pre skupiny
- Pridať priateľov pomocou QR kódu
- Pop-out message box pre čítanie a odosielanie správ
- Časová os a domovská stránka ako funkcie na iOS a Android
- Prístup cez viac osobných počítačov (Windows alebo Mac OS)

[22]

1.4 Push-To-Talk aplikácie (Walkie Talkie)

Push-to-talk, tiež známy ako Press-to-Transmit, je metóda komunikácie na half-duplex komunikačnej linke, zahŕňajúca obojsmerné rádio, kde pomocou tlačidla momentálneho prepnutia meníme režim pre príjem a prenos.

Push-to-talk v celulárnej sieti (PoC) je možnosť služby pre mobilné telefónne siete, ktorá umožňuje účastníkom používať svoje telefóny ako vysielačky s neobmedzeným rozsahom. Typické pripojenie PTT spája takmer okamžite. Významnou výhodou PoC/PTT je schopnosť jednej osoby dosiahnuť aktívneho hovoru skupiny s jediným stlačením tlačidla.

Voxer Walkie Talkie

Voxer Walkie Talkie je živý systém "push-to-talk" a systém hlasových správ. Správy vo Voxer sú dodávané hneď ako sú zaznamenávané a potom odosielané. Hlasové správy fungujú rovnako. Aplikácia funguje na Androide, Windows Phone a operačných systémov iOS. Voxer bol postavený na Node.js, Riak a Redis.

Node.js je softvérová platforma pre škálovateľné server-side a sieťové aplikácie. Node.js aplikácie sú napísané v JavaScripte. Node.js aplikácie sú navrhnuté tak, aby mali maximálnu priepustnosť a efektivitu, používajúc neblokujúce I/O a asynchrónne udalosti.

Výhody Voxer:

- Komunikácia s užívateľmi na Android a iPhone
- Posielanie audia, textu, fotografií a informácií o polohe
- Prehrávanie správ neskôr - všetky sú zaznamenané

- Chat s jedným užívateľom alebo so skupinou
- Vytvorenie správy aj offline
- Upozornenia na nové správy
- Pripojenie cez Wi-Fi, 3G, 4G, EDGE
- Prehrávať hlasové správy zrýchlene (2x alebo 3x rýchlosť)
- Spojenie sa s priateľmi na Facebooku

[23]

Zello PPT

Zello je ďalšia PTT aplikácia ktorá dokáže premeniť telefón alebo tablet na vysielačku. Funguje medzi systémami Android, iPhone, BlackBerry aj na stolných PC.

- Real-time streamovanie hlasu vo vysokej kvalite
- Verejné a súkromné kanály pre až 1000 používateľov
- Možnosť namapovať PTT (Push to talk) tlačidlo
- Podpora Bluetooth headset
- Hlasové histórie
- Upozornenia na volania
- Push notifikácia
- Práca cez Wi-Fi, 2G, 3G, 4G alebo mobilné dátové

[24]

Ďalšie aplikácie pre prenos multimediálnych dát

BBM, Facebook Messanger, TiKL Touch Talk Walkie-Talkie, HeyTell, Kakao Talk, Prip,...

2 Základy sieťovej komunikácie

2.1 Wi-Fi

Wi-Fi je pojem ktorým označujeme skupinu štandardov pre bezdrôtové lokálne siete LAN, v súčasnosti fungujúci podľa štandardu IEEE 802.11. Pôvodne bolo Wi-Fi vytvorené pre bezdrôtové zariadenia a lokálne siete, v dnešnej dobe sa ale bežne využíva pre pripojenie k internetu. Wi-Fi využíva bezlicenčné frekvenčné pásmo a vďaka tomu je ideálna pre vytváranie výkonných a finančne nenáročných sietí.

Bežná Wi-Fi môže mať jeden alebo viac prístupových bodov (Access Point), na ktoré sa môžu pripájať klientske zariadenia. Prístupový bod posiela svoje SSID (Service Set Identifier, teda sieťové meno) prostredníctvom paketov ktoré v tomto prípade voláme majáky (beacons), tie sa posielajú asi každých 100 ms rýchlosťou približne 1 Mbps. Tým je zaručené, že klienti ktorý prijímajú signál z AP, môžu komunikovať minimálnou rýchlosťou 1 Mbps. Podľa nastavení AP sa potom klient môže rozhodnúť či sa na neho pripojí. V prípade že má klient v dosahu viaceré prístupové body s rovnakým SSID, môže sa pripojiť na ten s lepšou silou signálu. Podľa štandardu Wi-Fi, kritériá pripojenia, roaming, alebo prechod medzi hotspotmi, sú prenechané na klientovi. V praxi teda každý bezdrôtový adaptér dosahuje rôzne výkony. [6]

V budúcnosti budú bezdrôtové adaptéry viac riadené operačným systémom. Hoci sa Wi-Fi prenáša vzduchom, má rovnaké vlastnosti ako neprepínaný ethernet. Dokonca sa môžu objaviť aj kolízie podobne ako v neprepínaných ethernetových sieťach. 802.11 používa ako metódu prístupu na médium povinne DCF (CSMA/CA).

2.2 Socket

Základom prenosu dát pomocou Wi-Fi siete je Socket a komunikácia medzi viacerými socketmi. Sieťový socket je koncový bod medziprocesového komunikačného toku naprieč počítačovou sieťou. Väčšina komunikácie medzi zariadeniami v dnešnej dobe je založená na Internetovom Protokole a preto aj väčšina sieťových socketov sú sieťové sockety.[11]

Socket API, rozhranie pre programovanie aplikácií poskytované zvyčajne operačným systémom, umožňuje aplikačným programom kontrolovať a používať sieťové sockety. API internetových socketov sú zvyčajne založené na Barkeleyho socketovom štandarde.

Socketová adresa je kombinácia IP adresy a čísla portu, podobne ako pri telefónnom pripojení kombinácia telefónneho čísla a konkrétneho rozšírenia. Na základe tejto adresy sockety doručujú prichádzajúce dátové pakety konkrétnemu aplikačnému procesu alebo vláknu.[11]

Okrem adresy je internetový socket charakterizovaný aj protokolom ako napríklad TCP alebo UDP. Kombinácia rovnakého čísla portu ale iného protokolu vytvorí rôzne sockety. Teda socket pracujúci s protokolom TCP na porte 80 je iný socket ako ten pracujúci na UDP taktiež s portom 80.

Server môže obsluhovať niekoľko klientov súčasne. Pre každého klienta vytvorí vlastný socket, pričom každý tento socket má z pohľadu servera rovnakú lokálnu socketovú adresu, ale rôzne vzdialené adresy pre každého klienta.

V rámci operačného systému a aplikácie ktorá socket vytvorila, je socket reprezentovaný unikátnou číselnou hodnotou ktorá sa nazýva socket descriptor. Operačný systém presmeriava užitočné dáta z prichádzajúcich IP paketov na odpovedajúcu aplikáciu pomocou extrahovania informácií o socketovej adrese z hlavičiek IP a transportných protokolov a odstránením hlavičky z aplikačných dát.[11]

Typy Socketov

Dostupných je niekoľko druhov internetových socketov:

Datagramové sockety – tiež známe ako nespojové sockety, ktoré sa využívajú pri UDP protokole

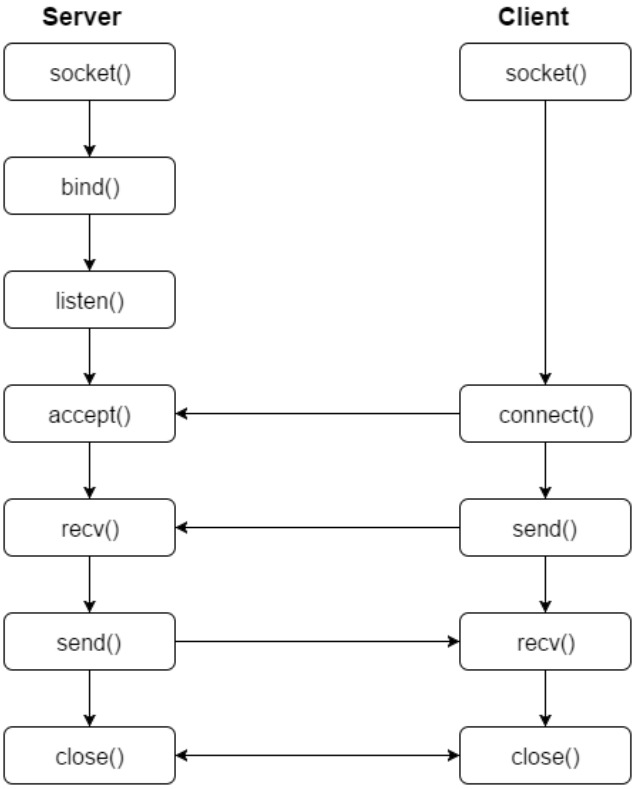
Streamové sockety – tiež známe ako spojovo orientované sockety ktoré vyžívajú TCP alebo STCP (Stream Control Transmission Protocol) protokoly

Surové (raw) sockety – typicky používané v routeroch a ostatných sieťových zariadeniach, tu je sieťová transportná vrstva vynechaná a hlavičky paketov sú sprístupnené aplikácii.

2.3 Socketová komunikácia

Pod pojmom socketová komunikácia rozumieme prepojenie dvoch alebo viacerých socketov podľa modelu Client-Server. Ako server sú označované počítačové procesy ktoré poskytujú aplikačné služby. Vo všeobecnosti akákoľvek aplikácia môže vytvoriť socketový server. Väčšinou sa ako server označuje strana komunikácie ktorá dáta prijíma. Server potom čaká na pripojenie socketov z klientskych programov = počúva na určenom porte. Na jeden server socket sa môže súčasne pripojiť niekoľko klientskych socketov a server s každým nadviaže TCP spojenie. Viacnásobná komunikácia však musí byť vytvorená na rovnakom porte. Server pre každé nadviazané spojenie vytvára socket, s lokálnym portom a lokálnou IP adresou, ktorý potom slúži vlastnému klientskemu procesu.[11]

Priebeh socketovej komunikácie na TCP protokole znázorňuje obrázok č.4



Obrázok č. 4 TCP klient-server model

Typické vytvorenie serverového socketu ktorý prijíma správu od klienta vyzerá na platforme Android takto:

```
ServerSocket serverSocket = new ServerSocket(PORT);
Socket client;
DataInputStream dis;
try
{
    client = serverSocket.accept();
    dis = new DataInputStream(client.getInputStream());
    dis.readUTF();
    ...

}
catch(IOException e) { System.out.println(e); }
```

Zdrojový kód č. 1: ukážka vytvorenia server socketu v TCP

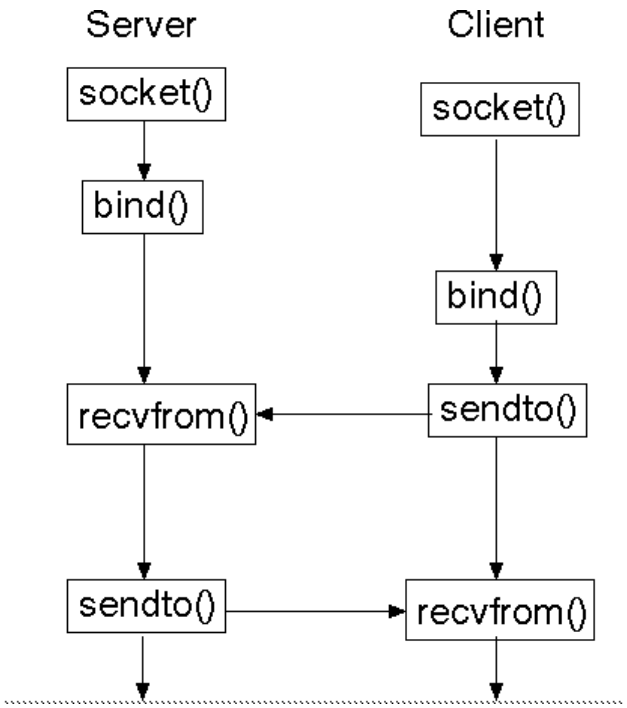
V tejto ukážke server používa na prečítanie správy prijatej od klienta triedu DataInputStream ktorá dovoľuje aplikácii čítať primitívne dátové typy Javy zo základného vstupného streamu. Na strane

klienta sa podobne použije trieda DataOutputStream ktorá naopak umožňuje posielanie primitívnych dátových typov. Ukážka jednoduchého klienta:

```
Socket client;
DataOutputStream dos;
try
{
    client = new Socket(IP_ADDRESS, PORT);
    dos = new DataOutputStream(client.getOutputStream());
    dos.writeUTF("1");
    ...

}
catch(IOException e) { System.out.println(e); }
```

Zdrojový kód č. 2: ukážka vytvorenia klient socketu v TCP

Komunikácia na protokole UDP vyzerá inak. Vzhľadom na to, že UDP je nespojový protokol, UDP socket nenaviaže spojenie, server nečaká kým sa na neho pripojí nejaký klient. UDP server nevytvára podriadené procesy pre každého klienta ale má len jeden proces ktorý zabezpečuje obsluhu dátových paketov zo všetkých klientov, postupne cez rovnaký socket. UDP sockety teda nie sú identifikované cez vzdialenú adresu, i keď ju každá správa má priradenú.[11]



Obrázok č. 5: UDP klient-server model

UDP sockety sú na platforme Android reprezentované triedou DatagramSocket. Posielanie dát medzi klientom a serverom je veľmi jednoduché. Server využívajúci UDP môže vyzerať napríklad takto:

```
DatagramSocket s = new DatagramSocket();
InetAddress address = InetAddress.getByName("IP_ADDRESS");
DatagramPacket p = new DatagramPacket(data, length, address, port);
s.send(p);

...
```

Zdrojový kód č. 3: ukážka vytvorenia server socketu v UDP

Klient nepotrebuje vedieť nič iné len číslo portu používaného serverom.

```
DatagramSocket s = new DatagramSocket(port);
byte[] buffer2 = new byte[length];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
s.receive(packet);

...
```

Zdrojový kód č. 4: ukážka vytvorenia klient socketu v UDP

2.4 Typy Wi-Fi komunikácie

Priame prepojenie cez lokálnu Wi-Fi sieť

Najbežnejší typ Wi-Fi komunikácie. Zariadenia sú pripojené na existujúcu Wi-Fi sieť, z ktorej potom preberajú lokálne IP adresy ktoré sú ďalej použité na vytvorenie socketovej komunikácie medzi nimi.

Mobilný hotspot

Hotspot je označenie pre fyzické miesto ktoré ponúka prístup k internetu cez Wi-Fi pripojenie lokálnej siete prostredníctvom routera pripojeného na ISP. Hotspoty zväčša využívajú Wi-Fi technológiu.

Mobilné hotspoty sú prenosné zariadenia alebo funkcie mobilných zariadení. Tieto zariadenia sa potom tvária ako prenosný bezdrôtový prístupový bod (portable Wi-fi AP) a router zároveň a umožňujú až piatim ďalším zariadeniam sa na ne pripojiť a komunikovať pomocou Wi-Fi siete alebo dokonca zdieľať ich internetové pripojenie. Väčšina operátorov má ale nastavený limit na mobilné pripojenie, takže zdieľanie takéhoto pripojenia určite nie je vhodné pre sťahovanie väčšieho objemu dát. Nevýhodou takéhoto pripojenia je aj energetická náročnosť a kratší dosah Wi-Fi ako u klasického

routeru. Je treba povedať že ak je mobilné zariadenie v stave prenosného hotspotu, tak nemôže byť pripojené na Wi-Fi. Nie všetky mobilné zariadenia sú však takouto funkciou vybavené. Operačný systém iOS podporuje tuto funkčnosť od verzie 4.2.5 a vyššie, alebo na iPhone 4, 4S, 5, iPad 3. generácie, zariadenia s Windows Phone 7, 8, 8.1 a samozrejme aj zariadenia s operačným systémom Android, ale je nutné povedať že to veľmi závisí od výrobcu a verzie.

Wi-Fi Direct

Wi-Fi Direct je štandard spočiatku nazývaný Wi-Fi P2P. Umožňuje veľmi jednoduché spojenie dvoch alebo viacerých zariadení bez nutnosti bezdrôtového prístupového bodu (Access Point). Komunikácia pomocou Wi-Fi Direct dosahujú rýchlosť ako pri používaní bežnej Wi-Fi siete. Jednou z výhod Wi-Fi Direct je to že dokáže prepojiť zariadenia ktoré sú od rôznych výrobcov. Ak je aspoň jedno zariadenie vybavené Wi-Fi Direct, je možné vytvoriť peer-to-peer spojenie a prenášať pomocou neho dáta. Wi-Fi Direct vytvára spojenie pomocou systému WPS (Wi-Fi protected setup). Tento systém priradzuje zariadeniam obmedzený prístupový bod. Spárovanie zariadení môže prebiehať rôzne. Napríklad pomocou signálu z Bluetooth, klasickým stlačením tlačidla alebo môže vyžadovať NFC (Near Field Communication).[7]

Podpora Wi-Fi Direct u dnešných zariadení je pomerne bežnou vecou. Google oznámil podporu Wi-Fi Direct u verzie Android 4.0. Niektoré iné zariadenia už ale mali túto funkčnosť implementovanú aj skôr, ako napríklad Samsung Galaxy S II s operačným systémom Android 2.3. V roku 2011 Google vydal Galaxy Nexus čo bolo prvé zariadenie s googlovskou implementáciou Wi-Fi Direct. U Blackberry je Wi-Fi Direct dostupná od verzie 10.2.1 a Apple zaviedol podporu tejto funkčnosti vo verzii iOS 8.[4]

3 Real Time Streamovanie v Androide

Väčšina mobilných zariadení v dnešnej dobe je vybavená hardwarom ktorý podporuje real-time streamovanie. OS Android poskytuje spôsoby vytvorenia real-time komunikácie. Bohužiaľ Android neposkytuje prístup k streamu z kamery pomocou SDK/NDK takže sa k (zakódovanému) streamu z kamery musíme dostať inými spôsobmi

3.1 MediaRecorder API

Jeden zo spôsobov je použitím MediaRecorderu ktorý slúži na nahrávanie audia a videa. Bohužiaľ MediaRecorder nepodporuje zapisovanie do bufferov. Video nahrávané MediaRecorderom je dostupné až po skončení nahrávania.

Typické použitie MediaRecorderu vyzerá takto:

```
MediaRecorder recorder = new MediaRecorder();
recorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
recorder.setVideoEncoder(MediaRecorder.VideoEncoder.H263);
recorder.setOutputFile(outputFile);
recorder.setPreviewDisplay(holder.getSurface());
recorder.prepare();
recorder.start();
// ...
recorder.stop();
```

Zdrojový kód č. 5: použitie MediaRecorder API

Musíme na MediaRecorder použiť niekoľko trikov. Pri bežnom použití MediaRecorderu sa používa zápis do súboru. Namiesto toho musíme vytvoriť LocalSocket a nastaviť zapisovanie do neho.

```
LocalServerSocket server = new LocalServerSocket("name");
receiver.connect(server.getLocalSocketAddress());
receiver.setReceiveBufferSize(BUFFER_SIZE);
sender = server.accept();
sender.setSendBufferSize(BUFFER_SIZE);

MediaRecorder recorder = new MediaRecorder();
recorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
recorder.setVideoEncoder(MediaRecorder.VideoEncoder.H263);
// podstrčiť náš falošný súbor
recorder.setOutputFile(outputFile);
recorder.setPreviewDisplay(holder.getSurface());
recorder.prepare();
recorder.start();
// ...
recorder.stop();
```

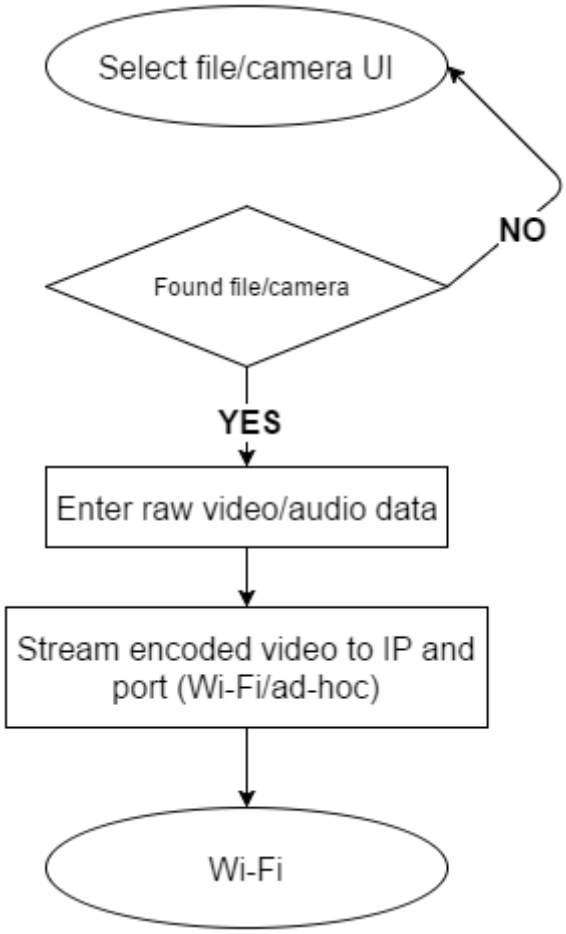
Zdrojový kód č. 6: použitie upraveného MediaRecorder API

Nastáva ale ďalší problém a to že MediaRecorder nám nepredá stream ale súbor, konkrétne súbor typu MPEG-4. MPEG4 je multimediálny formát tvorený skupinou kontajnerov ktoré obsahujú metadáta ako napríklad časové razítka, počet framov za sekundu, ale aj samotný audio/video stream. Musíme preskočiť všetky data pred 'mdat' kontajnerom ktorý obsahuje samotné dáta a tak dostaneme začiatok video streamu. [12]

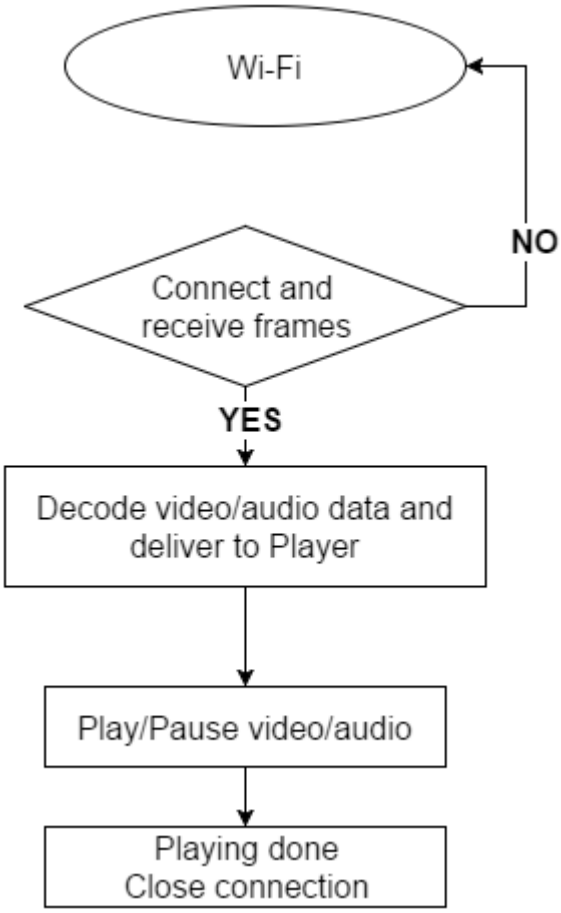
```
byte[] mdat = { 'm', 'd', 'a', 't' };
byte[] buffer = new byte[mdat.length];

do
{
    fillBuffer(buffer);
} while (!Arrays.equals(buffer, mdat));
```

Zdrojový kód č. 7: preskocenie mp4 mdat headeru

Ten musím potom zakódovať pomocou vybraného kodeku a takýto stream posielame do siete. Priebeh tohoto procesu znázorňuje obrázok č.6 a 7



Obrázok č. 6: posielanie videa, prebrané z [5]



Obrázok č. 7: prijímanie videa, prebrané z [5]

Získaný stream posielame pomocou vybraného protokolu, ako napríklad UDP, TPC alebo RTSP. Ničmenej pomocou tejto metódy real-time streamovania dochádza u výsledného obrazu k oneskoreniu

ktoré môže dosahovať až niekoľko sekúnd (2-3 sekundy na priemernej Wi-Fi), čo samozrejme závisí na kvalite Wi-Fi siete, pričom kvalita samotného obrazu taktiež nie je ideálna.

3.2 MJPEG pomocou protokolu HTTP

JPEG

JPEG je štandardná metóda stratovej kompresie používanej pre ukladanie počítačových obrázkov vo fotorealistickej kvalite. Formát súboru, ktorý tuto kompresiu používa, sa tiež bežne nazýva JPEG. Najrozšírenejšími príponami tohto formátu sú.jpg,.jpeg,.jfif,.jpe.

Skutočným názvom typu súboru je JFIF, čo znamená JPEG File Interchange Format. Skratka JPEG znamená Joint Photographic Experts Group, čo je vlastne konzorcium, ktoré túto kompresiu navrhlo.[14]

Keď sa bežne hovorí o súbore JPEG, myslí sa tým väčšinou súbor JFIF, alebo súbor Exif JPEG. Existuje však viacej formátov súborov založených na kompresii JPEG, napríklad JNG.

JPEG je vhodný pre fotografické snímky alebo maľby realistických scenérií s hladkými prechodmi v tóne a farbe. V tomto prípade funguje omnoho lepšie ako čisté bezstratové metódy, pričom poskytuje stále veľmi dobrú kvalitu obrazu. V skutočnosti poskytuje omnoho vyššiu kvalitu obrazu ako GIF, ktorý je síce bezstratový (a hodí sa na text a ikonky), ale vyžaduje veľmi silnú kvantizáciu pre plnofarebný obraz fotografie.[14]

JPEG využíva stratovú formu kompresie založenú na diskrétnej kosínusovej transformácií. Táto matematická operácia prevádza každý snímok videového zdroja z priestorovej domény do frekvenčnej domény. Percepčný model založený na ľudskom psychovizuálnom systéme vyradí vysokofrekvenčné informácie, ako ostré prechody intenzity a farbu odtieňa. V transformačnej doméne, spôsob redukcie informácií sa nazýva kvantizácia. Zjednodušene, kvantizácia je metóda pre optimálnu redukciu veľkej škály čísel (s odlišnými výskytmi každého čísla) do menšej a transformačná doména je vhodná reprezentácia obrazu pretože vysokofrekvenčné koeficienty, ktoré k celkovému obrazu prispievajú menej ako ostatné koeficienty, sú charakterizované malými hodnotami s veľkou kompresivitou. Kvantizačné koeficienty sú potom sekvenčne a bezstratovo zabalené do výstupného bitstreamu. [14]

Kompresná metóda je zvyčajne stratová, čo znamená že niektoré pôvodné informácie obrázku sú stratené a nedajú sa obnoviť a to môže mať dopad na kvalitu obrázku. Je dostupný aj voliteľný režim nestratovej kompresie definovaný v štandarde JPEG, ale nie je v produktoch veľmi podporovaný.

Motion JPEG

MJPEG (Motion JPEG) je formát videa, ktorý je najčastejšie používaný v digitálnych a IP kamerách. Každý snímok je komprimovaný samostatne podľa štandardu JPEG. MJPEG samotný

nemá oficiálny štandard. Implementuje ho však niekoľko kodekov - medzi nimi predovšetkým knižnica libavcodec z frameworku FFmpeg. Z absencie štandardu vyplývajú problémy možnej vzájomnej nekompatibility týchto kodekov. Všetky skomprimované snímky sú tu kľúčové, čo má výhodu rýchleho posunu vo videu. [13]

MJPEG je často používaný v nelineárnych systémoch pri strihaní videa. Moderné procesory sú dostatočne výkonné pre prácu s videami vo vysokom rozlíšení takže nie je potrebný žiaden špeciálny hardware. Podpora MJPEG je takisto rozšírená v prístrojoch ktoré sa používajú na nahrávanie a editáciu videa.

MJPEG je vnútro-framová (intraframe) kompresná schéma. Vzhľadom na to moderné vnútro-framové formáty ako napríklad MPEG1, MPEG2 alebo H.264/MPEG-4 AVC, dosahujú kompresného pomeru k reálnemu svetu 1:50 alebo lepšie, nedostatok medzi-framovej predikcie obmedzuje efektívnosť MJPEG na 1:20 alebo nižšie, v závislosti na tolerancii k priestorovým artefaktom v skomprimovanom výstupe. Pretože snímky sú komprimované nezávisle na sebe, MJPEG kladie nižšie požiadavky na pamäťové a procesné komponenty zariadenia.[14]

MJPEG je jednoduchý na implementovanie vďaka tomu že používa vyspelý kompresný štandard (JPG) spolu s dobre naprogramovanými knižnicami a jeho vnútro-framovou kompresnou metódou. Toleruje rýchlo sa meniaci pohyb vo video streamoch zatiaľ čo v kompresných schémach využívajúcich medzi-framovú kompresiu, sa môže často vyskytovať neprijateľná strata kvality keď sa výrazne mení obsah videa medzi každým snímkom.

Streamovanie cez HTTP

HTTP streamovanie rozdeľuje každý snímok MJPEG-u do samostatných HTTP odpovedí na zadané značky. RTP streamovanie vytvára pakety sekvencií JPEG obrázkov ktoré môžu byť prijímané klientskymi aplikáciami ako sú napríklad QuickTime alebo VLC.

Ako odpoveď na GET požiadavok na MJPEG súbor alebo stream, server posiela sekvenciu JPEG rámcov (framov) cez HTTP. Špeciálny typ MIME

multipart/x-mixed-replace;boundary=<nazov-hranice>

informuje klienta o tom že má očakávať niekoľko častí (framov) ako odpoveď ohraničenú <nazov-hranice>. Názov tejto hranice je zverejnený vnútri deklarácie tohoto typu MIME. TCP spojenie nie je ukončené kým klient chce prijímať nové obrázky a server ich môže poskytovať.

4 Požiadavky na vývoj aplikácie

Súčasťou tejto práce je aj vytvorenie vlastnej aplikácie ktorá bude pomocou Wi-Fi prenášať multimediálne dáta ako audio a video, poprípade údaje zo senzorov alebo statické dáta. Cieľom je ale vytvoriť aplikáciu ktorá umožní zariadeniam priame prepojenie, bez pomoci akejkoľvek formy online serveru. Takisto aplikácia bude zohľadňovať situácie keď nie je dostupná žiadna Wi-Fi sieť.

4.1 Funkčnosť aplikácie

Súhrn základných požiadavkou na aplikáciu:

1. **Prepojenie dvoch Android zariadení pomocou lokálnej Wi-Fi siete**.

   Užívatelia budú môcť vyhľadávať zariadenia na ktoré sa môžu pripojiť v rámci lokálnej Wi-Fi siete.

2. **Prepojenie dvoch Android zariadení pomocou WiFi-Direct**.

   Bude možné vyhľadávanie a pripojenie na zariadenia ktoré majú zapnuté WiFi-Direct. Zariadenia takto môžu komunikovať bez toho aby boli po blízku Wi-Fi siete.

3. **Vytvorenie mobilného HotSpotu**.

   Aplikácia bude schopná vytvoriť v zariadení prenosný HotSpot ktorý zariadeniam umožní komunikáciu pomocou Wi-Fi. Takisto bude aplikácia schopná vyhľadávať Wi-Fi siete a Hotspoty ktoré sú v dosahu zariadenia.

4. **Prenos audia a videa v reálnom čase**.

   Aplikácia bude schopná posielať dáta z kamery a mikrofónu v reálnom čase a zároveň sa bude automaticky prispôsobovať súčasným podmienkam. Cieľom je vytvoriť prenos s čo najlepším pomerom oneskorenia a kvality.

5. **Prenos údajov z kompasu zariadenia**.

   Aplikácia bude posielať údaje zo senzoru kompasu. Bude dostupná aj vizualizácia týchto dát.

6. **Prenos a nahrávanie multimediálnych dát**.

   Užívatelia budú môcť medzi prepojenými zariadeniami posielať aj klasické multimediálne dáta ako sú obrázky, videa, alebo audio súbory. Takisto bude možnosť priamo nahrať video/audio alebo spraviť fotku.

7. **Prenos textových správ (chat)**.

Prenos textových správ bude formou klasického chatu.

4.2 Software pre vývoj

Pre vývoj aplikácií na platformu Android sú v súčasnosti najlepšími možnosťami Eclipse alebo Android Studio.

Obrázok č. 8: Android studio

Android Studio je pomerne nové vývojové prostredie, ktoré sa stalo oficiálnym vývojovým prostredím pre vytváranie Android aplikácií. Bolo uvedené v 16.5.2013 na konferencii Google I/O. Android Studio je postavené na IntelliJ IDEA Community Edition založenom na Jave od spoločnosti Jetbrains. Je voľne dostupné pod licenciou Apache License 2.0 a je dostupné na platformy Windows, Linux aj Mac OS X. Android Studio je optimalizované čisto pre vývoj Android aplikácií s čím prichádzajú funkčnosti ktoré ostatným vývojovým prostrediam chýbajú alebo nie sú na takej úrovni. AS teda sľubuje vývojárom jednoduchšie vyvíjanie vďaka lepšej sade nástrojov. Google taktiež upozorňuje vývojárov že by mali prejsť na Android Studio.

Oproti Eclipsu zavádza Android Studio zmeny ako napríklad štruktúru projektu podľa systému Gradle, zmenu práce s manifest súbormi, namiesto konceptu Workspace zavádza moduly a ďalšie zmeny. Sú to zmeny na ktoré si vývojár musí zvykať, hlavne ak prechádza na Android Studio po dlhoročných skúsenostiach s Eclipsom.[4]

Obrázok č. 9: Eclipse

Android Studio by malo byť náhradou za Eclipse s rozšírením Anrdoid Development Tools (ADT). Eclipse, ktorý je rovnako voľne dostupný, open source software vydaný pod licenciou Eclipse Public License. Je takisto dostupný na všetkých populárnych operačných systémoch, je to však komplexnejšie vývojové prostredie ktoré sa nepoužíva len na vývoj Android aplikácií ale aj ako vývojové prostredie pre ďalšie jazyky ako napríklad Java, C, C++, PHP, Python alebo ďalšie. Táto aplikácia bola vyvíjaná na verzii Eclipse 4.4 Luna.

Pre vývoj Android aplikácií v Eclipse je však potrebný plugin Android Development Tools. Tento plugin rozširuje možnosti Eclipsu o vytváranie a nastavovanie Android projektov a takisto ponúka vizuálne náhľad na užívateľské rozhranie. Do štandardného editora Eclipsu ADT taktiež pridáva editor XML ktorý pomáha vytvárať a upravovať súbory pre menu, layout alebo manifest.

Pre vývoj aplikácie ktorá je súčasťou tejto práce som použil Eclipse verziu Luna. Hlavným dôvodom bol nedostatok skúsenosti s vývojom v Android Studiu. Naopak čo sa Eclipsu týka nie je to môj prvý projekt a tak som dobre oboznámený s možnosťami tohoto vývojového prostredie a nebol som spomalený prispôsobovaním sa na nové prostredie a funkčnosť.

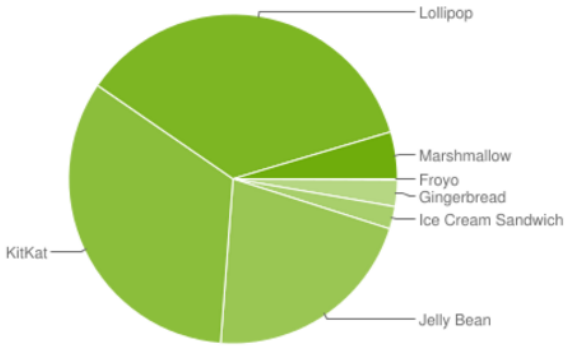
4.3 Výber verzie Androidu

Ako minimálna verzia OS Android pre beh vyvíjanej aplikácie bola zvolená verzia Android 4.0 Ice Cream Sandwich (API level 14). Táto verzia predstavila radu nových funkcií ako napríklad near-field communication (NFC), vylepšený prehliadač, nový manažér kontaktov s integrovanou podporou sociálnych sietí, prístup ku kamere a ovládaniu audia pri zamknutej obrazovke, monitorovanie a obmedzovanie využívania dát a mnohé ďalšie.

Obrázok č. 10: Android Ice Cream Sandwich

Android 4.0 je prvá verzia Androidu ktorá zaviedla aj podporu WiFi-Direct, čo je jednou zo základných požiadavkou na aplikáciu. Jedným z hlavných dôvodov prečo bola 4.0 zvolená ako minimálna verzia je fakt, že v súčasnosti väčšina zariadení ktoré bežia na OS Android, má verziu 4.0 alebo vyššiu. Je teda zbytočné podporovať aj verzie Androidu ktoré už dnes nemajú zastúpenie ani 5%. Vyplýva to z prieskumu dát na Google Play Store.[4]

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%



Obrázok č. 11: graf verzií Androidu na trhu, prebrané z [4]

Dáta použité v prieskume sú aktuálne k dátumu 4.4.2016.

Graf na obrázku č.11 nám jasne ukazuje že verzie staršie ako 4.0 tvoria už len malé percento u súčasných používateľov. Na grafe nie sú zohľadnené verzie staršie ako 2.2, avšak už podľa prieskumov v roku 2013, verzie staršie ako Android 2.2 zaberali menej ako 1% zariadení ktoré sa pripájali na servere Google. Prieskumy ukazujú že v súčasnosti je najrozšírenejšou verziou Android 4.4 KitKat. Verzia 2.2 Froyo už podľa očakávania prakticky nemá zastúpenie. Najnovšia verzia Android 5.0 bola vydaná 7.8.2014 a má zatiaľ len 3.3% kvôli krátkej dobe na trhu a kvôli obmedzenému množstvu zariadení ktoré túto verziu doposiaľ používajú

4.4 Testovacie zariadenia

Vývoj aplikácie ktorá ktorej kľúčové funkcie sa týkajú, ako v prípade tejto práce, napríklad Wi-Fi, prenosu obrazu a multimédií, alebo využívania senzorov je veľmi komplikované a v niektorých prípadoch proste nemožné na emulátore ADT. Na otestovanie takýchto funkcií sú potrebné reálne zariadenia. Takisto je nutné povedať že spúšťanie Android aplikácií, alebo debuggovanie je na reálnych zariadeniach oveľa rýchlejšie ako na emulátore.

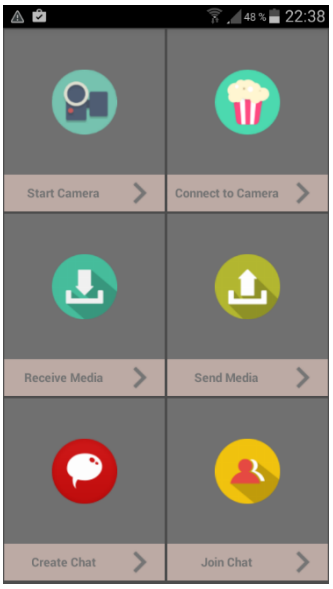
Na odladenie funkčnosti vyvíjanej aplikácie boli použité tieto zariadenia:

- Samsung S4 Mini
- Samsung Galaxy Note 10.1
- Samsung S3 Mini
- Google Nexus 10

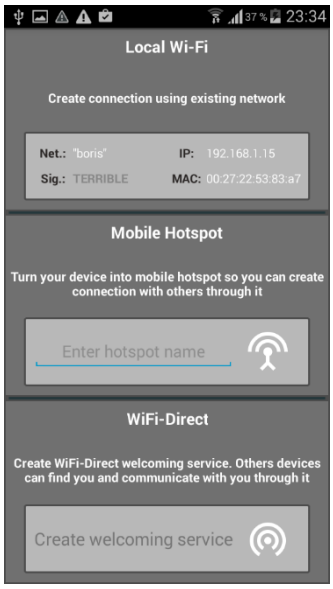
Všetky zariadenia majú OS Android 4.x, podporu Wi-Fi, WiFi-Direct, kompas, na všetkých zariadeniach je predná aj zadná kamera, a každé má funkciu zapnutia mobilného hotspotu. Software aj hardware všetkých zariadení je teda dostačujúci pre účely tejto aplikácie.

5 Analýza a návrh

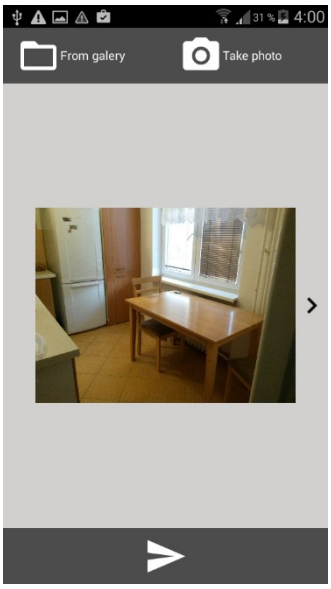
Výsledná aplikácia by mala byť schopná poskytovať požadovanú funkčnosť v čo najmenšom počte krokov, kvôli jednoduchosti používania. Cieľová funkčnosť aplikácie je rozvrhnutá medzi niekoľko hlavných obrazoviek, ktoré sú zobrazené na nasledujúcich obrázkoch:



Obrázok č. 12: úvodná obrazovka

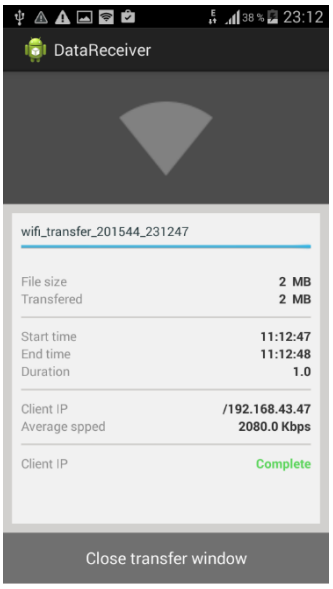


Obrázok č. 13: výber typu pripojenia



Obrázok č. 14:výber dát na poslanie

Obrázok č. 15: chat



Obrázok č. 16: prenos dát

Obrázok č. 17: audio/video prenos

V úvodnej obrazovke si užívateľ môže zvoliť s akými dátami chce pracovať. Teda či chce vytvoriť real-time prenos audia a videa, posielať len statické dáta ako napríklad obrázky alebo audio súbory, alebo chce len posielať textové dáta, teda chat. U všetkých týchto možností si ešte musí vybrať či chce vytvoriť komunikáciu alebo sa pripojiť ako klient.

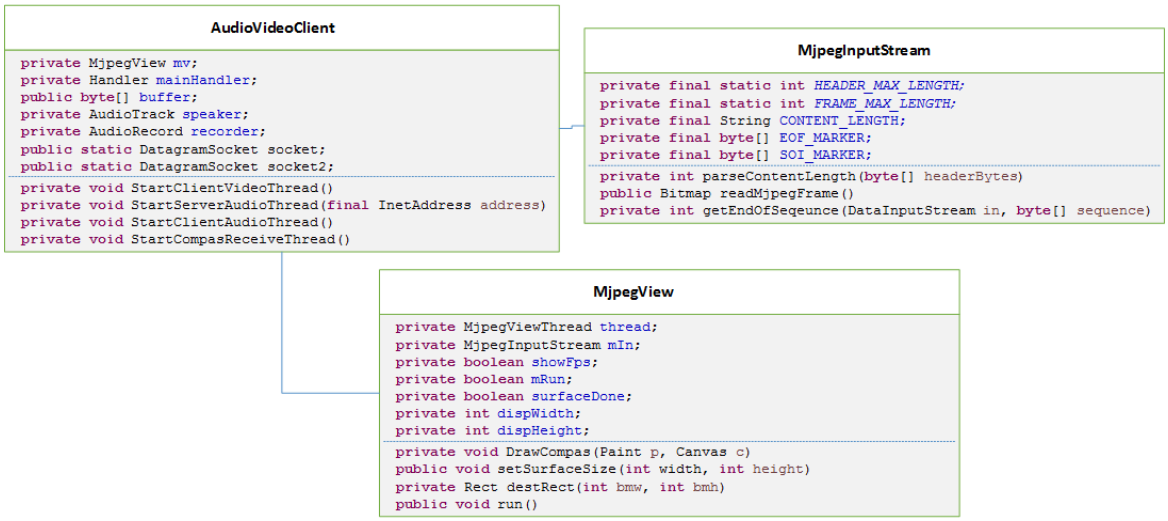
Postup vytvorenia real-time prenosu audia a videa:

Užívateľ1: Spustí aplikáciu, vyberie možnosť Start Camera (vytvoriť novú kameru), vyberie si druh pripojenia na ktorom prenos bude fungovať a čaká na pripojenie klienta

Užívateľ2: Spustí aplikáciu, vyberie Connect to Camera (pripojiť sa k existujúcej kamera), spustí vyhľadávanie zariadení a pripojí sa na existujúcu kameru

O vytvorenie novej kamery a posielanie audio a video streamu sa stará trieda AudioVideoServer, ktorá vytvára jednotlivé vlákna na posielanie/prijímanie audia a videa. Táto trieda si po spustení vlákna pre posielanie videa vytvára inštanciu triedy CameraPreview, v ktorej prebieha zachytávanie snímkov z kamery a posielanie pomocou HTTP protokolu. V tejto triede prebieha aj samotná dynamická úprava kvality posielaných snímkov. Na strane klienta sa o pripojenie k existujúcej kamere stará trieda AudioVideoClient ktorá podobne ako AudioVideoServer spúšťa vlákna pre posielanie/príjem audio a video streamu. Táto trieda počas príjmu videa pracuje s triedou MjpegInputStream ktorá sa stará o parsovanie hlavičky a dát prijatých cez HTTP. Výsledné dáta potom vracia do AudioVideoClient kde sú zobrazované pomocou triedy MjpegView. Triedny diagram vyzerá nasledovne (kvôli prehľadnosti obsahuje triedny diagram na obrázku nižšie len základné premenné a metódy, reálne triedy obsahujú oveľa viac):

Obrázok č. 18: triedny diagram pre server real-time komunikácie



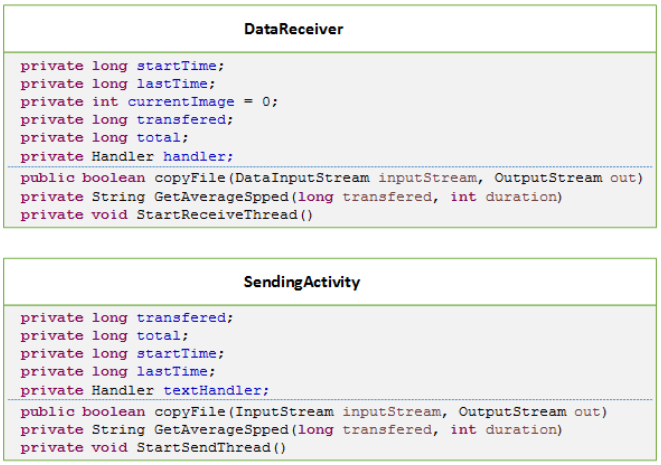
Obrázok č. 19: triedny diagram pre klienta real-time komunikácie

Postup prenosu statických dát:

Užívateľ1: Spustí aplikáciu, vyberie možnosť Receive Data (prijímať statické dáta), vyberie si druh pripojenia na ktorom prenos bude fungovať a čaká na pripojenie klienta, sleduje priebeh prenosu

Užívateľ2: Spustí aplikáciu, vyberie možnosť Send Data (posielať statické data), vyberie si typ dát a buď si vytvorí nové (odfotí/nahrá/natočí), alebo vyberie existujúce zo zariadenia, potom spustí vyhľadávanie a pripojí sa na užívateľa čakajúceho na príjem dát, sleduje priebeh prenosu

Príjem dát je implementovaný v triede DataReceiver. Tu prebieha vytvorenie socketu ktorý čaká na klientsku aplikáciu ktorá sa na neho pripojí a zaháji tak prenos. Podobne sa na strane klienta o pripojenie na server a samotný prenos dát stará trieda SendingActivity. Tá najskôr po vytvorení spojenia pošle informácie o prenášanom súbore, ako napríklad typ, veľkosť, názov a potom začne s prenosom samotného súboru. DataReceiver a SendingActivity (opäť len to najdôležitejšie):



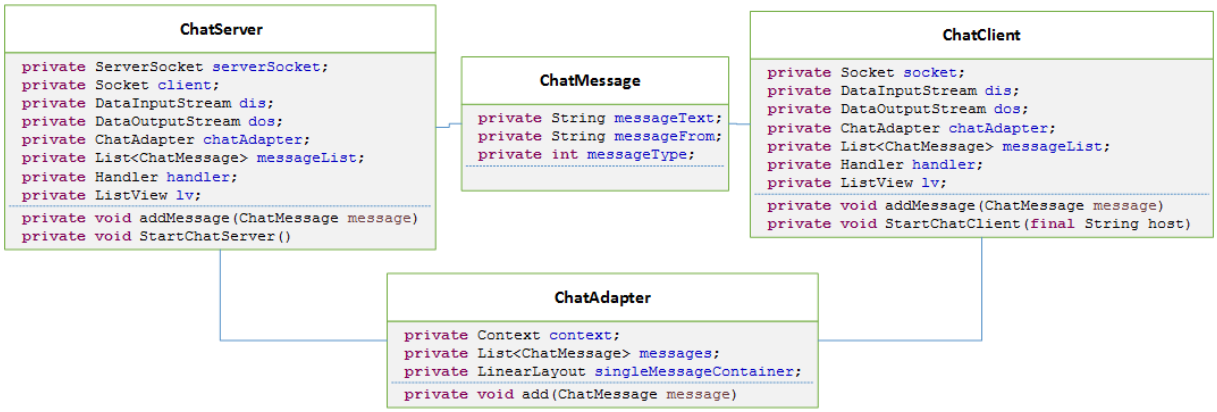
Obrázok č. 20: triedy pre výmenu dát

Postup prenosu textových dát (chat):

Užívateľ1: Spustí aplikáciu, vyberie možnosť Start Chat (vytvorí nový chat), vyberie si druh pripojenia na ktorom prenos bude fungovať a čaká na pripojenie klienta, komunikuje s užívateľom2

Užívateľ2: Spustí aplikáciu, vyberie možnosť Join Chat (pridať sa do existujúceho chatu), potom spustí vyhľadávanie a pripojí sa na užívateľa ktorý má vytvorený chat, komunikuje s užívateľom1

Vytvorenie a pripojenie k chatu opäť zabezpečujú dve podobné triedy a to ChatServer a ChatClient. ChatServer sa podľa očakávania stará o vytvorenie socketu ktorý čaká na klienta. Po vytvorení sa obom užívateľom otvorí rovnaké užívateľské prostredie kde budú vidieť svoju konverzáciu. O zobrazovanie nových správ sa stará trieda ChatAdapter ktorá okrem aktualizovania nových správ aj farebne obom užívateľom rozlišuje správy podľa ich autora. Nové správy sú vytvárané pomocou triedy ChatMessage ktorá obsahuje základné údaje o správe. Triedny diagram (len dôležité údaje):



Obrázok č. 21: triedny diagram pre chat

6 Vývoj aplikácie

Vývoj aplikácie mal dve základné fázy. V prvej fáze som preskúmaval existujúce aplikácie na priame prepojenie Android zariadení. Zisťoval som akým spôsobom majú implementovanú komunikáciu a ako posielajú mediálne dáta v reálnom čase. Vytvoril som si niekoľko menších aplikácií na ktorých som samostatne testoval jednotlivé funkcie ktoré mala cieľová aplikácia obsahovať. Dôvod prečo som vytváral viacero menších aplikácií bolo testovanie. V tejto fáze vývoja bola pre mňa väčšina funkčnosti niečo, s čím som pracoval prvý krát, takže bolo jasné že prvá verzia aplikácie by určite nebol finálnou. Z môjho pohľadu bolo teda oveľa jednoduchšie otestovať určitú funkčnosť na primitívnej aplikácii, ktorá je zameraná len na danú funkčnosť, ako mať zložitú aplikáciu, v ktorej by testovanie bolo zložitejšie.

Prvý krok bolo vytvoriť prepojenie dvoch zariadení. Vyskúšal som si naprogramovať jednoduchú klient-server komunikáciu pomocou lokálnej Wi-Fi a posielať statické dáta. Vyskúšal som viaceré open-source knižnice a aplikácie ako napríklad *libstreaming*, *spydroid-ipcamera* alebo *android-eye* ktoré sú zamerané na prenos videa a audia pomocou Wi-Fi. Na základe týchto príkladov som sa najskôr rozhodol implementovať prenos dát v reálnom čase implementovaním RTSP protokolu ktorý by pomocou MediaRecorder API posielal zakódovaný stream z kamery. Toto riešenie som ale nakoniec opustil. Implementácia tohoto riešenie bola pomerne zložitá a výsledky ktoré sa mi podarilo dosiahnuť neboli dostatočné. Preto som sa rozhodol vyskúšať metódu prenosu MJPEG. Na základe testov na stránke https://foxdogstudios.com/peepers som sa rozhodol pre implementáciu MJPEG pomocou HTTP protokolu. Vyskúšal som si implementovať jednoduchú verziu takéhoto typu prenosu videa v reálnom čase. Výsledky boli oveľa lepšie čo sa týka kvality obrazu a oneskorenia prenosu, ako pri prvom pokuse. Keďže cieľová aplikácia mala byť schopná okrem videa prenášať aj audio, naprogramoval som samostatný prenos audia medzi zariadeniami a spustil som ho paralelne s MJPEG prenosom.

Druhá fáza vývoja tejto aplikácie už bol vývoj samotnej cieľovej aplikácie. Bolo to vlastne skladanie jednotlivých menších aplikácií v ktorých som mal implementovanú určitú funkčnosť do jedného celku. Hlavným cieľom bolo vytvoriť aplikáciu ktorá umožní priame prepojenie dvoch zariadení s OS Android a umožní posielať audio/video poprípade iné statické dáta. Vývoj aplikácie postupoval podľa toho:

1. Vyhľadávanie zariadení na sieti – v rámci tohoto kroku bolo implementované aj vyhľadávanie dostupných prístupových bodov, vytváranie Hotspotu na danom zariadení a WiFi-Direct komunikácia

2. Posielanie audia a video z kamery a mikrofónu zariadenia – implementovanie MJPEG streamu cez HTTP protokol a posielanie audio streamu

3. Posielanie statických dát a dát zo senzorov zariadenia – posielanie statických mediálnych dát a údajov z kompasu. V tomto kroku bol pridaný aj jednoduchý chat

6.1 Vyhľadávanie zariadení

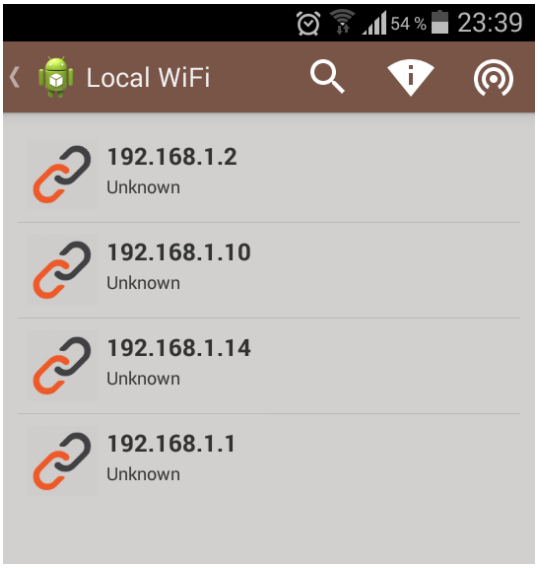
Komunikácia medzi zariadeniami by bola veľmi nepraktická keby sme pri každom pokuse o spojenie museli zadávať IP adresu odznova. Preto je v aplikácií implementovaný sken lokálnej siete ktorý zobrazí aktívne zariadenia na sieti. Obrazovka pre vyhľadávanie zariadení obsahuje menu s troma tlačidlami:

Obrázok č. 22: screenshot aplikačného menu

Tlačidlá zľava:

1. Spustí vyhľadávanie aktívnych zariadení na sieti
2. Otvorí vyhľadávanie dostupných sietí a hotspotov
3. Otvorí vyhľadávanie WiFi-Direct

Po spustení vyhľadávania sa zobrazí prehľadný zoznam zariadení ktoré boli dosiahnuté. Zobrazí sa aj ich IP adresa ale užívateľ ju nemusí nikde prepisovať, stačí ak klikne na vybraté zariadenie a aplikácia sa pokúsi na danú IP pripojiť.



Obrázok č. 23: screenshot výsledku vyhľadávania network scanneru

Mobilný prístupový bod

Ak klientske zariadenie nie je na rovnakej sieti ako server na ktorý sa chce pripojiť, môže si dať v rámci aplikácie vyhľadať dostupné prístupové body (Access Point) v okolí a pripojiť sa na ne.

Android dovoľuje aplikáciám prístup k zobrazeniu stavu Wi-Fi pripojení až do veľmi nízkej úrovne. Pri vytváraní mobilného prístupového bodu musíme na zariadení vypnúť Wi-Fi, na čo potrebujeme mať v manifest súbore aplikácie povolenia na prístup a zmenu stavu Wi-Fi:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

Aplikácie majú prístup takmer ku všetkým informáciam o pripojení. Na toto poskytuje Android WifiManager API, ktoré slúži na obsluhu všetkých aspektov Wi-Fi pripojenia. Inštanciu triedy WifiMinager vytvoríme pomocou metódy getSystemService:

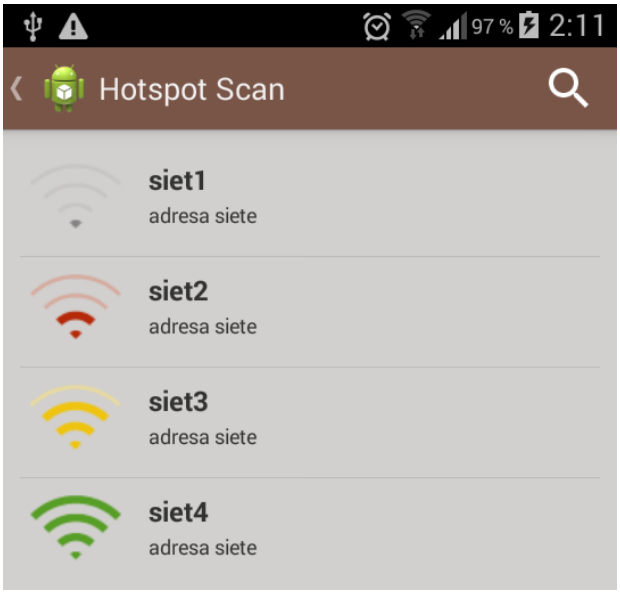
```
WifiManager mainWifi =
(WifiManager)getActivity().getSystemService(Context.WIFI_SERVICE);
```

Ako ďalší krok si musíme zaregistrovať BroadcastReceiver ktorý bude vykonávať sken Wi-Fi sietí a hotspotov. Jednoduchý kód pre vytvorenie takéhoto receiveru môže vyzerať takto:

```
class WifiReceiver extends BroadcastReceiver
{
    public void onReceive(Context c, Intent intent)
    {
        StringBuilder sb = new StringBuilder();
        final List<ScanResult> wifiList;

        wifiList = mainWifi.getScanResults();

        for(int i = 0; i < wifiList.size(); i++)
        {
            String SSID = wifiList.get(j).SSID;
            String BSSID = wifiList.get(j).BSSID;
            String level = String.valueOf(wifiList.get(j).level);
            addDevice(new WifiNetworkInfo(SSID, BSSID, level));
        }
    }
}
```

Zdrojový kód č. 8: broadcast receiver pre vyhľadávanie Hotspotov

U každého nájdeného prístupového bodu si aplikácia uloží jeho SSID, BSSID a level. SSID je vlastne názov siete. BSSID je adresa siete. Na základe levelu sa určuje sila signálu daného prístupového bodu. Aplikácia takto podľa sily signálu aj vizuálne rozlišuje nájdené prístupové body na ktoré sa potom užívateľ môže pripojiť.



Obrázok č. 24: screenshot výsledku vyhľadávania Hotspotov

Okrem toho že aplikácia vie vyhľadávať a pripájať sa na prístupové body, vie si aj vytvoriť vlastný. Zjednodušený kód pre vytvorenie a základné nastavenie hotspotu môže vyzerať napríklad takto:

```
WifiConfiguration netConfig = new WifiConfiguration();
netConfig.SSID = name;
netConfig.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.OPEN);
netConfig.allowedProtocols.set(WifiConfiguration.Protocol.RSN);
netConfig.allowedProtocols.set(WifiConfiguration.Protocol.WPA);
netConfig.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
try
{
    …
    netConfig =
    (WifiConfiguration)getWifiApConfigurationMethod.invoke(wifiManager);
}
catch (Exception e) { … }
```

Zdrojový kód č. 9: vytvorenie Hotspotu

Bohužiaľ nie všetky zariadenia ponúkajú takúto možnosť, ale väčšina moderných smartfónov alebo tabletov už touto funkčnosťou vybavená je. To je prvé riešenie prípadu keď potrebujeme prepojiť zariadenia keď v dosahu nie je žiadna existujúca Wi-Fi sieť.

WiFi-Direct komunikácia

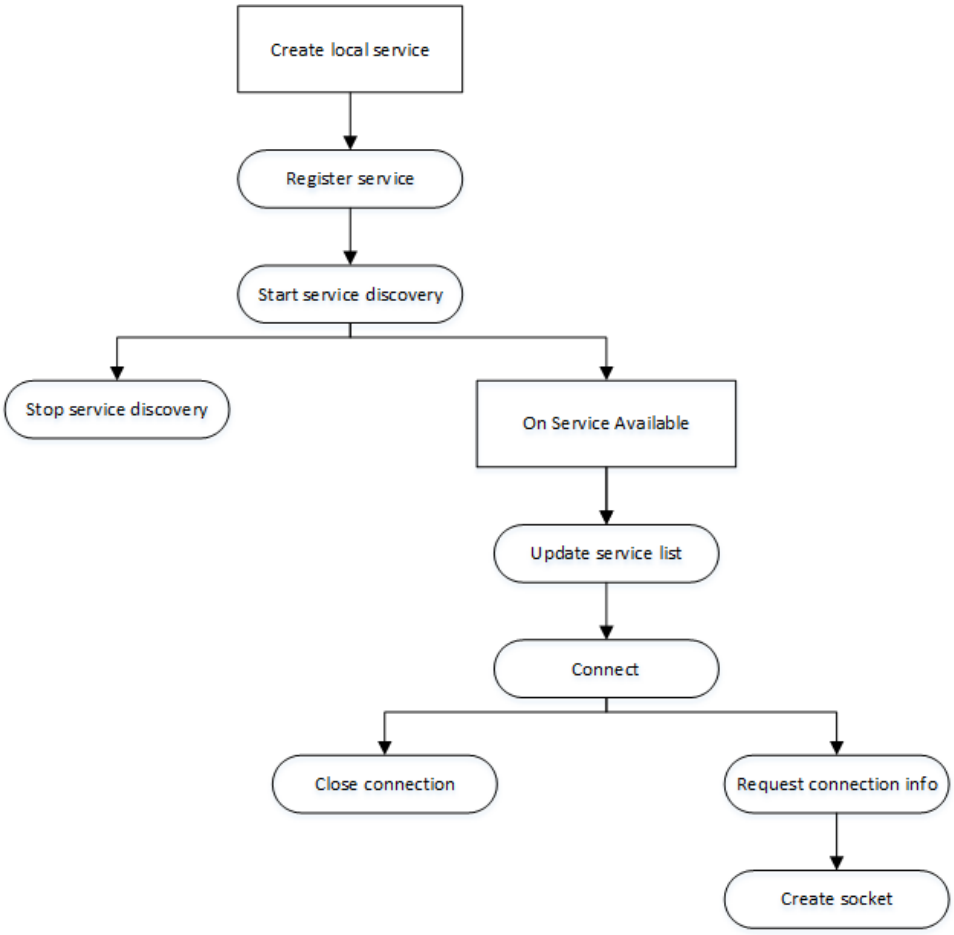
Komunikácia prostredníctvom technológie WiFi-Direct prináša niekoľko problémov. Na základe skúseností pri vývoji tejto aplikácie a na základe skúseností ostatných programátorov ktoré som našiel na internete je treba povedať že technológia WiFi-Direct ešte nie je bezchybná. Asi najvážnejší problém okolo WiFi-Direct počas vývoja bolo, že po nesprávnom ukončení aplikácie, ako napríklad pri páde aplikácie po neodchytenej chybe, sa občas stalo že WiFi-Direct prestala na zariadení fungovať a zariadenie sa muselo reštartovať. Ďalším zvláštnym chovaním WiFi-Direct je určovanie takzvaného GroupOwnera. Aplikácia sa na základe toho kto je GroupOwner rozhoduje ktorú časť kódu spustí. O tom kto sa stane GroupOwnerom rozhoduje parameter groupOwnerIntent. Ten môže nadobúdať hodnoty od 0 po 15 pričom 0 je najmenšia šanca že sa stane GroupOwnerom a 15 naopak najvyššia. Android ale určuje tento parameter pri vytvorení komunikácie náhodne, takže ak potrebujeme nastaviť aby zariadenie ktoré napríklad chce vysielať data bolo vždy určené ako GroupOwner, musíme mu pri vytváraní manuálne nastaviť hodnotu groupOwnerIntent na 15. To je fakt o ktorom sa väčšina tutoriálov a dokumentácií pre WiFi-Direct na Android nezmieňuje.

Pre vytvorenie WiFi-Direct komunikácie potrebujeme v aplikačnom manifeste niekoľko povolení a pre prácu s Wi-Fi:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
```

WiFi-Direct komunikáciu, čo je vlastne Peer-to-Peer komunikácia, môžeme prevádzkovať ako na existujúcej Wi-Fi sieti, tak aj bez nej. V tejto aplikácii potrebujeme aby komunikácia fungovala aj mimo dosah Wi-Fi siete takže ako prvé je nutné vytvoriť lokálnu službu ktorá bude viditeľná pre ostatné zariadenia aj keď v dosahu nebude žiadna existujúca Wi-Fi sieť. Ak chceme poskytovať lokálnu službu, musíme ju zaregistrovať pre vyhľadávanie služieb. Akonáhle je služba zaregistrovaná, Android automaticky reaguje na požiadavky na vyhľadávanie služieb od peerov. Ďalší krok je vytvorenie metódy na vyhľadávanie služieb, alebo teda na počúvanie prichádzajúcich záznamov. To zaobstará funkcia WifiP2pManager.DnsSdTxtRecordListener. Na získanie informácií o službe si musíme vytvoriť WifiP2pManager.DnsSdServiceResponseListener ktorý dostáva informácie o popise spojenia. Nakoniec už len stačí nastaviť hodnoty groupOwnerIntent kôli tomu aby aplikácia vedela, či zariadenie chce posielať alebo prijímať dáta.

Priebeh vyhľadávania a spojenia zariadení pomocou WiFi-Direct môžeme vidieť na diagrame na obrázku č.15

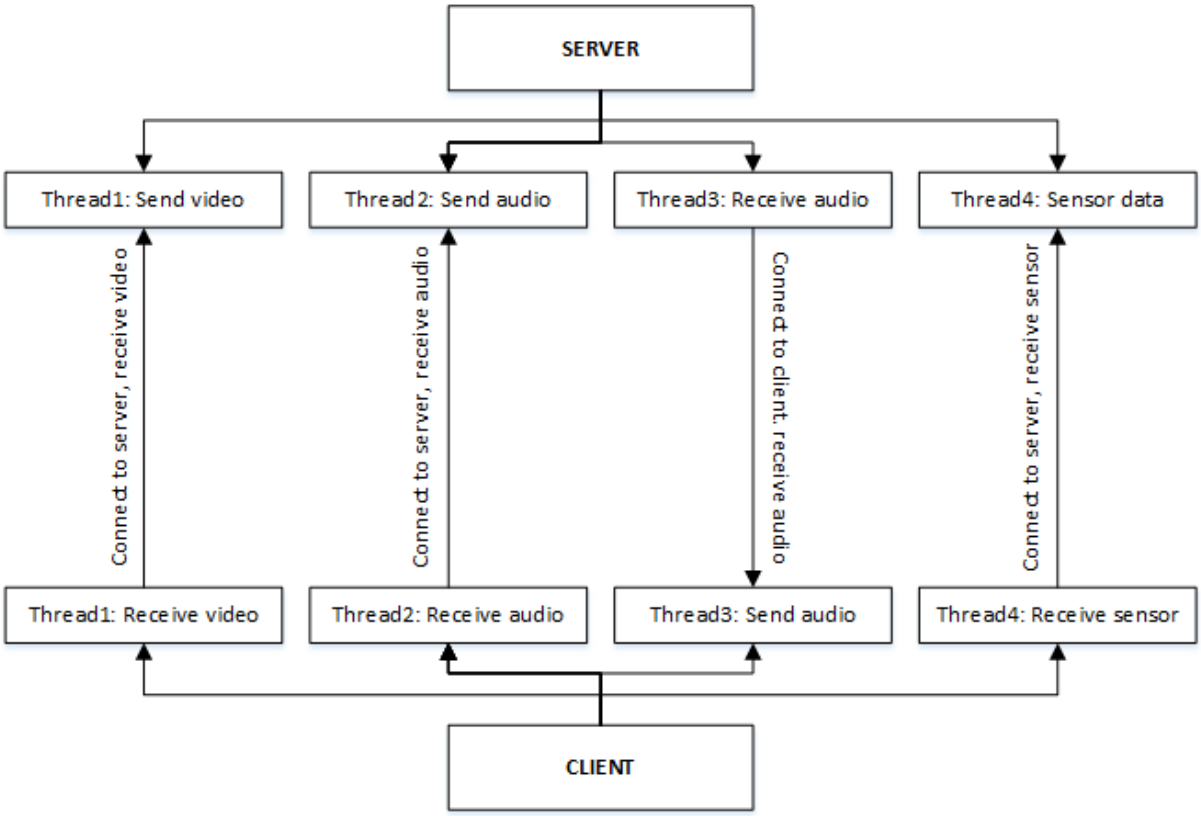


Obrázok č. 25: priebeh WiFi-Direct vyhľadávania a spojenia

WiFi-Direct je tak ďalšou variantou komunikácie medzi zariadeniami bez nutnosti existujúcej Wi-Fi siete. Bohužiaľ ani táto metóda nemusí byť podporovaná všetkými zariadeniami, ale opäť väčšina moderných zariadení už WiFi-Direct podporuje.

6.2 Prenos audia a videa

Implementácia prenosu audia a videa v reálnom čase bola rozdelená do dvoch krokov. Prvým krokom bolo implementovať prenos videa a druhým dosadiť k nemu prenos audia. To sa deje v samostatných vláknach. Aplikácia funguje tak že zariadenie ktoré v komunikácií vystupuje ako server začne posielať v samostatných vláknach zároveň aj MJPEG stream, aj stream s audiom. Keď sa na server pripojí klient a začne prijímať dáta, server si zistí jeho IP adresu a začne prijímať audio z klienta. Tento proces môžeme zjednodušene naznačiť obrázkom č.16:



Obrázok č. 26: priebeh prenosu audio a video streamu

Video je prenášané ako MJPEG pomocou HTTP, takže na strane zariadenia ktoré vysiela stream s obrázkami z kamery bolo nutné vytvoriť funkčnosť ktorá neustále zachytáva obrázky z kamery, obalí ich HTTP hlavičkou s potrebnými informáciami o danom obrázku a odošle do streamu. Android našťastie obsahuje udalosť onPreviewFrame ktorá zachytáva každý zobrazený frame z kamery a vráti ho ako bytové pole. Keďže chceme posielať MJPEG tak je nutné takýto frame pred začiatkom posielania prekonvertovať na formát JPG. Defaultne totižto táto metóda vracia formát NV21. Ak máme obrázok v správnom formáte, ďalším krokom je doplnenie HTTP hlavičky a odoslanie. Ukážka posielania do streamu:

```
stream.write(("HTTP/1.0 200 OK\r\n" +
              "Server: iRecon\r\n" +
              "Connection: close\r\n" +
              "Content-Type: multipart/x-mixed-replace;" +
              "boundary=" + boundary + "\r\n" +
              "\r\n" +
              " + boundary + "\r\n").getBytes());
stream.write(("Content-type: image/jpeg\r\n" +
              "Content-Length: " + buffer.size() + "\r\n" +
              "X-Timestamp:" + ts + "\r\n" +
              "\r\n").getBytes());

buffer.writeTo(stream);
```

```
stream.write(("\r\n--"+AudioVideoServer.boundary+"\r\n").getBytes());
stream.flush();
```

Zdrojový kód č. 10: posielanie framu cez HTTP

V tejto ukážke sa do streamu zapíše jeden frame ktorý je uložený v premennej buffer vo forme bytového poľa. V hlavičke je dôležitý hlavne atribút Conten-Type: multipart. Ten značí že sa posielajú viaceré subjekty. Jednotlivé subjekty sú oddelené takzvanou hranicou (boundary). Hranica je definovaná ako riadok ktorý sa skladá z dvoch znakov "-" a z parametra boundary definovanom v hlavičke.

Na posielanie audio streamu je v aplikácií použitý protokol UDP. Ten sa na posielanie audio streamu hodí viac ako protokol TCP pretože je rýchlejší a u audia nám prakticky nevadí, ak sa stane že sa niektorý paket nedoručí ako by mal. Prenos audia sa spúšťa paralelne s prenosom videa, vo vlastnom vlákne. Posledná časť ktorú server vykonáva je prijímanie audio streamu z klienta, ktoré začne v momente keď sa klient pripojí a server tak pozná jeho IP adresu.

Na strane prijímajúceho zariadenia, teda klienta, je implementovaný HTTP klient ktorý sa stará o prijímanie MJPEG streamu. HTTP klient nerobí nič iné, len to že sa pripojí na server a preberá dáta. Pripojenie na server je pomocou URL v tvare: http://SERVER_IP:PORT. Z prijatých dát si potom postupne vytvára jednotlivé framy ktoré prezentuje na obrazovku. Prijímanie a posielanie audio streamu je podobný proces ako na strane servera. Klient sa paralelne pripojí na server z ktorého začne prijímať audio a video dáta a zároveň spustí vlákno ktoré začne posielať vlastný audio stream.

Čo sa oneskorenia týka tak MJPEG dosahuje skvelé výsledky dosahujúce oneskorenie maximálne pol sekundy. Záleží to samozrejme na kvalite Wi-Fi ale predpokladáme aspoň priemerné podmienky. Problém môže nastať pri rapídnom poklese kvality kedy sa musí upraviť kvalita prenosu. Môže sa stať že prenos na krátku dobu zamrzne, ale keď sa znova rozbehne nedochádza k zvýšeniu oneskorenia pretože MJPEG sa nám nesnaží vykresliť všetky oneskorené obrázky, ale skočí rovno na aktuálny.

6.3 Dynamická úprava kvality obrazu

Jedným z hlavných požiadavkou aplikácie je že kvalita obrazu sa má automaticky prispôsobovať aktuálnym podmienkam. Tento problém je v aplikácií riešený dvoma spôsobmi. Za prvé kvalita prenášaného framu z kamery. Počas prevodu framu z pôvodného NV21 formátu na JPG je možnosť nastaviť kvalitu kompresie. Čím nižšia kvalita kompresie, tým rýchlejší je prenos framov. Aplikácia začína s kvalitou kompresie 70. Počas prenosu videa si zaznamenáva počet prenesených framov za sekundu (FPS) prenášaného obrazu. Počas prenosu sa každé tri sekundy spustí funkcia QualitySetter ktorá na základe FPS a mení kvalitu obrazu, teda rozhoduje či je nutné znížiť kvalitu kompresie alebo naopak zvýšiť. Na základe testovaní som zistil že optimálna hranica kedy sa obrázky posielajú natoľko

plynulo, že človek nepozná že sa nejedná o video, je okolo 15 framov za sekundu. Podľa toho si funkcia na úpravu kvality obrazu prispôsobuje silu kompresie. Najnižšia možná kvalita obrazu je rovná 10. Kvalita obrazu by síce išla nastaviť až na minimum rovné 1, ale to už nemá zmysel. Kvalita 10 je už veľmi nízka. Na základe toho sú defaultne nastavené parametre:

```
// pocitadlo FPS
private int FPS = 0;
// pocet FPS po 1 sekunde, na zaciatku nastaveny rovnako ako FPS_LIMIT aby
cez prve kolo cyklu presiel normalne
private int LAST_FPS_COUNT = 15;

// timestamp z predošlého času FPS - niečo ako predchadzaju sekunda v
ktorej sa odoslali framy, používa sa na zistenie či sa nové framy posielaju
už aspoň 1 sekundu
private long LAST_FPS_TIME = 0;

// odsledovaný idealný FPS je 15
private int FPS_LIMIT = 15;

// kvalita posielaného obrazu, na začiatku nastavená na 70
private int IMAGE_QUALITY = 70;

// timestamp pre 3-sekundove časové kolo na prenastavenie kvality  obrazu
private long ROUND_TIME_STAMP;

private void QualitySetter()
{
    // ak je posledne FPS mensie ako 13 a sucasna kvalita obrazu je  aspon
    20, tak znizi kvalitu o 10
    if (LAST_FPS_COUNT < 13 && IMAGE_QUALITY >= 20)
    {
        IMAGE_QUALITY -= 10;
    }
    // inak pozera ci nie je FPS dost vysoke (viac ako 20) na to aby mohol
     zvysit kvalitu obrazu o 10
    else if (LAST_FPS_COUNT > 17 && IMAGE_QUALITY <= 90)
    {
        IMAGE_QUALITY += 10;
    }
    // nova timestamp pre casove kolo na kvalitu obrazu
    ROUND_TIME_STAMP = System.currentTimeMillis()/1000;
}
```

Zdrojový kód č. 11: nastavovanie kvaly obrazu u MJPEG

Ďalšie obmedzenie prenosu spočíva v množstve prenesených framov za sekundu. Ak prenosová rýchlosť dostatočne vysoká a dosahuje viac ako 15 framov za sekundu, ďalšie framy sa v rámci jednosekundového kola neprenesú pretože je to zbytočné. Ak už pri hodnote FPS približne 15 ani nepoznáme rozdiel medzi videom a skupinou za sebou idúcich obrázkov, tak je zbytočné prenášať v rámci jedného sekundového kola viac ako 15 framov. Pri každom prenose framu sa teda spustí funkcia FpsCounter ktorá meria počet prenesených framov za sekundu a na základe toho rozhoduje či počas danej sekundy má posielať ďalšie framy. Funkcia ale rozhoduje aj o veľkosti hranice prenášaných

framov. Ak je pripojenie dostatočne kvalitné a FPS sa drží na vysokej úrovni tak funkcia navyšuje hranicu pre prenášané FPS. Ak naopak je nízke FPS tak hranicu znižuje. Hodnota 15 je ale najnižšia hranica pre FPS. Samotná funkcia je ukázaná na zdrojovom kóde č. 12:

```
private void FpsCounter(long tsLong)
{
    FPS++;  // pocitadlo pre vypis FPS
    if (LAST_FPS_TIME == 0)    // prve kolo
    {
        FPS = 0;
        LAST_FPS_TIME = tsLong;
    }
    // ak je cas noveho framu rozdielny od toho posledneho
    else if (tsLong > LAST_FPS_TIME)
    {
        int diff = (int) (tsLong - LAST_FPS_TIME);    // casovy rozdiel
        if (diff != 0)    // ak je rozdiel aspon sekunda
        {
            // ak je po sekunde pocitadlo framov vacsie ako obmedzenie+1
            tak navysi obmedzenie
            if (FPS > FPS_LIMIT+1)
            {
                FPS_LIMIT++;
            }
            // ak je po sekunde pocitadlo mensie o 2 tak znizi obmedzenie
            else if (FPS-1 < FPS_LIMIT)
            {
                FPS_LIMIT--;
            }
            // pocet framov v tejto sekunde ... aby som potom na zaciatku
            novej sekundy vedel kolko bolo v starej
            LAST_FPS_COUNT = FPS;
        }
        // nakoniec resetne pocitadlo a cas
        FPS = 0;
        LAST_FPS_TIME = tsLong;
    }
}
```

Zdrojový kód č. 12: upravovanie prenosu na základe FPS

Aplikácia potom na základe parametrov FPS_LIMIT a FPS rozhoduje, či má ďalšie framy prenášať alebo nie. Rozhoduje o tom jednoduchá podmienka:

```
if (FPS_LIMIT < 15 && FPS > FPS_LIMIT)
{
    Log.d("FPS", "neprenasa");
}
else
{
    …
}
```

6.4 Prenos statických a senzorových dát

Implementácia prenosu statických dát zahŕňala prenos multimediálnych dát ako sú fotky, alebo audio a video súbory. Užívateľ si môže buď vybrať z existujúcich súborov v zariadení alebo má možnosť priamo v aplikácií si fotky/audio/video vytvoriť. V rámci tohoto bodu bol do aplikácie pridaný aj jednoduchý klient-server chat.

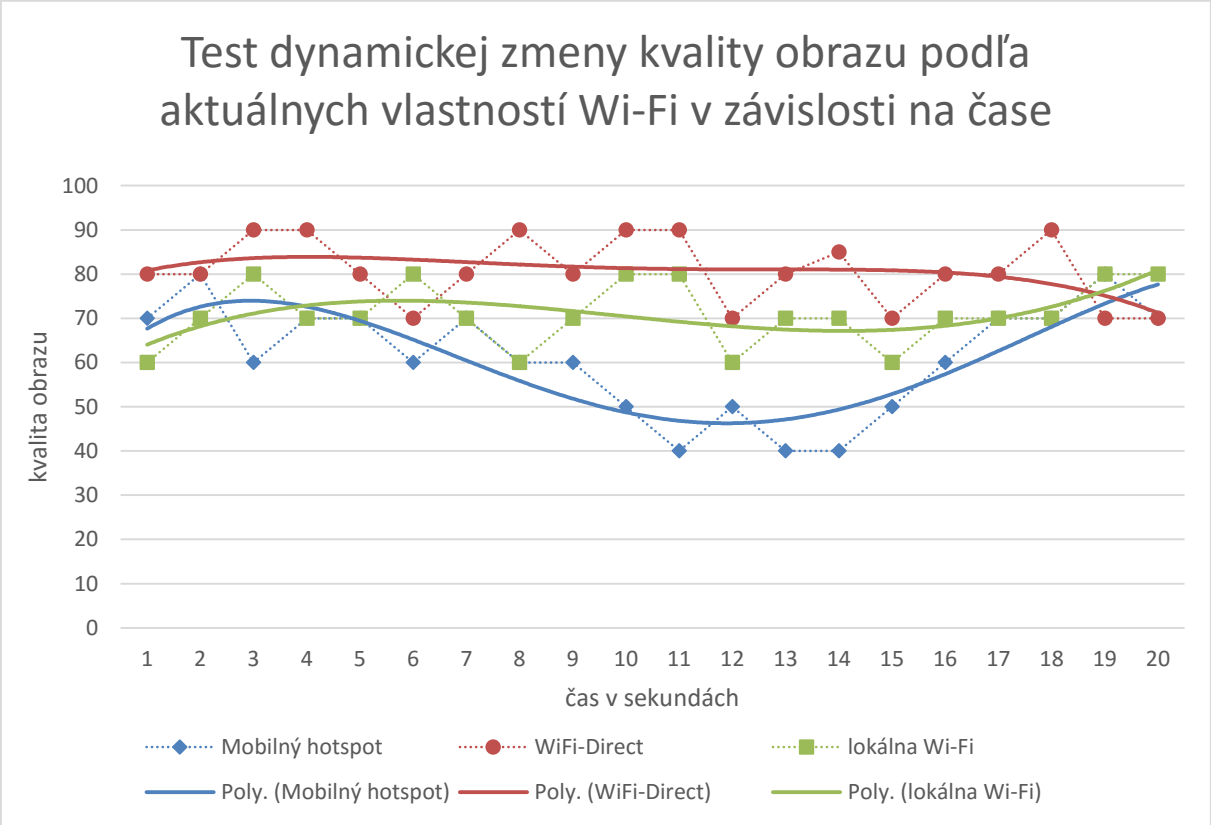
V rámci senzorových dát sa posielajú údaje z kompasu zariadenia. Funkčnosť bola doplnená do obrazovky s prenosom audio/video streamu. Klient tak vidí okrem MJPEG obrazu aj dáta zo senzoru v podobe kompasu ktorý sa natáča podľa smeru do ktorého je serverové zariadenie natočené.

Obrázok č. 27: ukážka kompasu

7 Testovanie

Testovanie výslednej aplikácie spočívalo v zhodnotení kvality prenosu, hlavne video streamu. Boli vykonané viaceré testy ktoré mali ukázať ako sa prenos videa mení na základe určitých faktorov. Testy boli vykonané pre všetky typy spojenia.

Cieľom prvého testu bolo ukázať kvalitu spojenia pomocou jednotlivých druhov spojenia. Test prebiehal v rámci jednej budovy, pričom zariadenia boli v rôznych miestnostiach, teda oddelené stenami. Výsledok je ukázaný na grafoch:



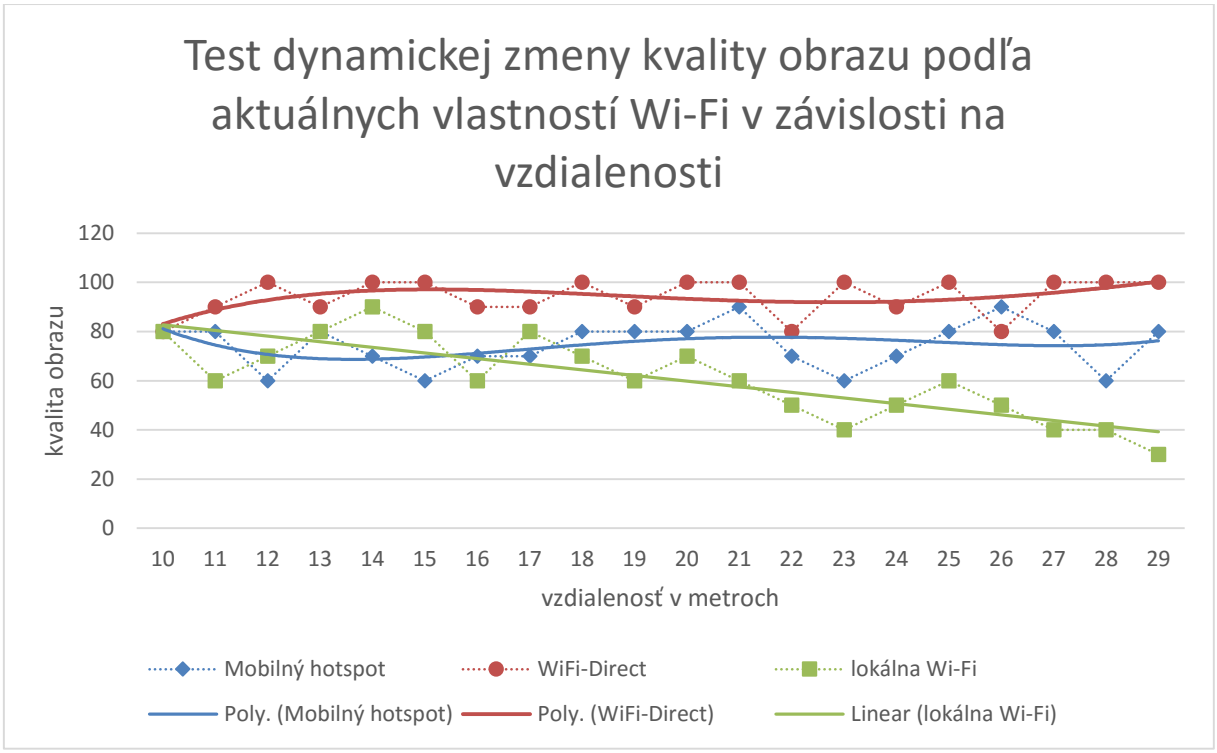
Obrázok č. 28: Graf kvality obrazu pre Wi-Fi spojenia

	min FPS	max FPS	priemer	min kvalita	max kvalita	priemer
Wi-Fi Direct	2	28	15	70	90	80
lokálna Wi-Fi	3	24	17	60	80	70
Hotspot	10	26	14	40	80	60

Tabuľka č. 1: výsledné dáta testovania v závislosti na čase

Tento test ukázal že na krátke vzdialenosti dosahuje najlepších výsledkov WiFi-Direct. Test bol niekoľko krát opakovaný a výsledky sa vždy o niečo líšili, ničmenej WiFi-Direct dosahoval vždy najlepšiu kvalitu obrazu. Samozrejme prenos na existujúcej Wi-Fi závisí hlavne od kvality danej siete, ale aj tak dosahoval veľmi dobré výsledky. Wi-Fi na ktorej boli tieto testy vykonané dosahovala rýchlosti približne 900 kilobajtov za sekundu. Čo sa FPS týka, tak pri všetkých typoch pripojenia a aj pri rôznych hodnotách kvality obrazu boli veľké výkyvy ktoré naznačujú že intenzita signálu Wi-Fi sa neustále menila aj na malej vzdialenosti. Najstabilnejšie FPS sa ukázalo byť na lokálnej Wi-Fi.

Ďalší test bol zameraný na kvalitu prenášaného videa v závislosti na vzdialenosti zariadení. Vzdialenosti v tomto teste určite nie sú presné ale mali by byť brané skôr ako približné hodnoty. Test bol vykonávaný na otvorenom priestranstve. Testovaná kvalita bola meraná do vzdialenosti približne 30 metrov, pričom na začiatku boli zariadenia vzdialené približne 10 metrov a každé tri sekundy bolo zariadenie vzdialené o približne meter. Výsledky sú opäť viditeľné na grafoch:



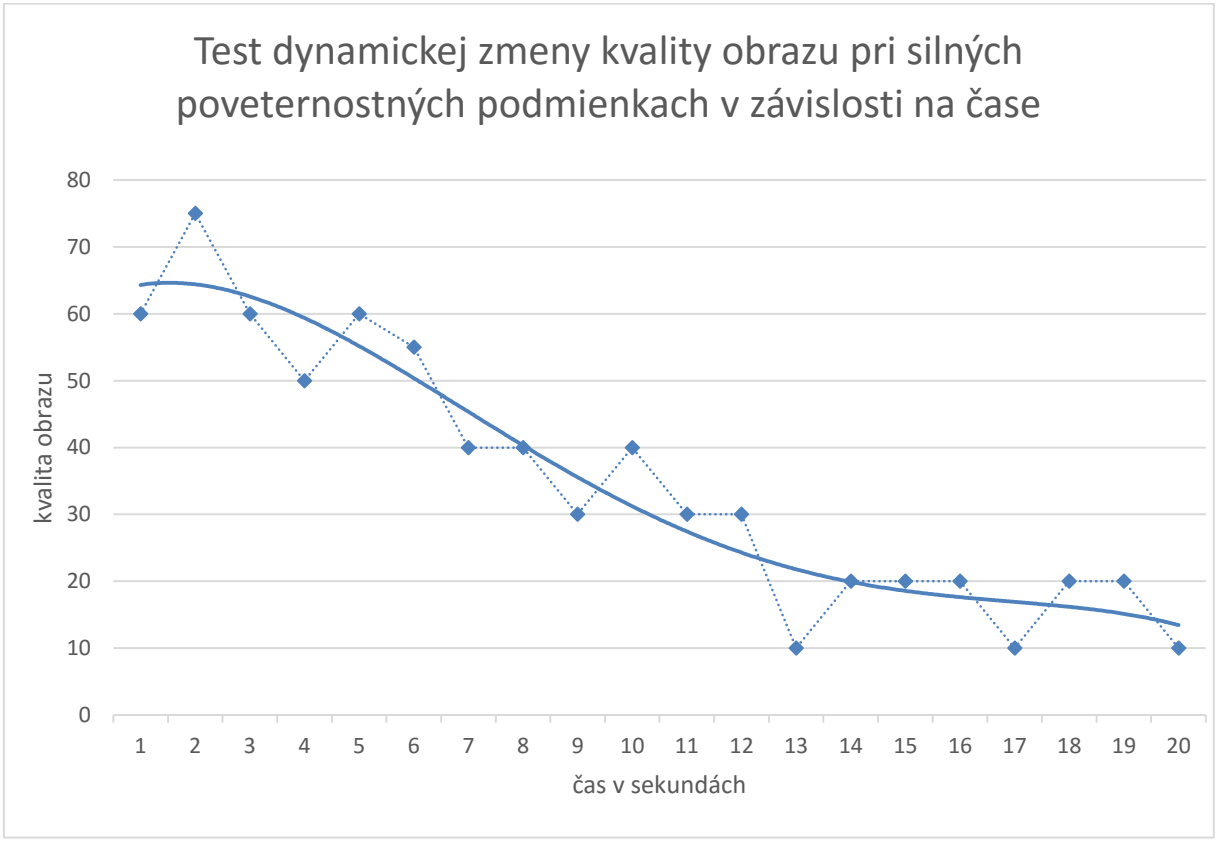
Obrázok č. 29: test kvality obrazu v závislosti na vzdialenosti

	min FPS	max FPS	priemer	min kvalita	max kvalita	priemer
Wi-Fi Direct	7	26	16	80	100	90
lokálna Wi-Fi	1	28	17	30	90	60
Hotspot	8	23	16	60	90	70

Tabuľka č. 2: výsledné dáta testovania v závislosti na vzdialenosti

Aj tento test bol opakovaný niekoľkokrát. Test vzdialenosti opäť dopadol najlepšie pre WiFi-Direct. Lokálna Wi-fi ale aj v tomto teste bola limitovaná dosahom routeru, u ktorého sa po približne 30 - 35 metroch signál strácal úplne. Zaujímavý výsledok mal mobilný hotspot ktorý aj v rámci testovanej vzdialenosti dosahoval veľmi dobé výsledky. Opakované testovanie ukázalo že vzdialenosť kedy ešte signál z mobilného hotspotu dosahuje, je približne necelých 40 metrov. WiFi-Direct bolo samostatne testované na vzdialenosť približne 50 metrov a bol stále dosiahnuteľný. Výkyvy v FPS sú opäť veľké z čoho vyplýva že aj na otvorenom priestranstve a pri väčšej vzdialenosti dochádza ku značnému kolísaniu signálu.

Zaujímavý faktor pri testovaní bol vietor. Náhodným testovaním som zistil že vietor ovplyvňuje intenzitu signálu. Počas silného vetra sa mi podarilo otestovať prenos video pomocou mobilného hotspotu a WiFi-Direct. Výsledky oboch prenosov boli veľmi podobné a tak som na základe výsledkov z oboch typov pripojenia vytvoril nasledujúci graf ktorý znázorňuje klesajúcu kvalitu obrazu pri silom vetre:



Obrázok č.30: test kvality obrazu v pri silnom vetre

8 Záver

Cieľom tejto práce bolo preskúmať možnosti priamej komunikácie na zariadeniach s OS Android pomocou technológie Wi-Fi a navrhnúť vlastnú aplikáciu ktorá by dokázala vytvoriť takéto spojenie a prenášať multimediálne dáta, hlavne audio a video v reálnom čase. Pre dosiahnutie tohoto sa bolo najskôr potreba oboznámiť so základnými pojmami týkajúcimi sa multimédií a streamovania na platforme Android a oboznámiť sa s niektorými protokolmi ktoré sa najčastejšie používajú pri prenosoch multimédií. Súčasťou práce bolo ale aj preskúmať už existujúce aplikácie ktoré majú podobnú funkčnosť a prípadne sa inšpirovať niektorou existujúcou alebo sa naopak vyvarovať nedostatkom ktoré obsahuje.

V priebehu vývoja tejto aplikácie boli popísané aj rôzne spôsoby komunikácie a posielania dát v rámci Wi-Fi, ktoré boli v konečnom dôsledku zakomponované do výslednej aplikácie. Ukázali sme si aj súčasné možnosti real-time streamovania na platforme Android a na základe porovnaní vybrali možnosť ktorá nám viac vyhovovala. Nasledovne bol v práci naznačený postup samotného vývoja, aplikácie. Boli zmienené niektoré problémy alebo chyby ktoré nastali pri vývoji ale aj spôsoby akými boli riešené.

Na konci tejto práce boli prezentované aj niektoré testovacie dáta na ktorých sa dá zhodnotiť kvalita prenosu výslednej aplikácie a takisto zmeny v prenose pri rôznych typoch pripojenia a za rôznych okolností.

Výsledná aplikácia spĺňa požadovanú funkčnosť a momentálne je v stave kedy sa s ňou dá pracovať. Prenos audia a videa bol otestovaný a kvalita obrazu rovnako ako aj oneskorenie sú vo výsledku podľa očakávaní dobré. Ďalším cieľom je dostať aplikáciu na Google Play, kde by sa lepšie ukázalo či je o takýto typ aplikácií záujem. Predtým ale bude potreba popracovať na dizajne aplikácie a poprípade pridať pár "vychytávok" ktoré by uľahčili používanie, alebo zlepšiť niektoré súčasné funkcie, ako napríklad zlepšiť vyhľadávanie zariadení aby dokázalo vyhľadať len zariadenia na ktorých je spustená služba ktorú hľadáme, to znamená posielanie broadcastových správ o existencii služieb ktoré klienti očakávajú. Ako ďalšie by sa v budúcnosti mohlo dopracovať pripojenie cez NFC aby sa dalo zariadenia spárovať bez vyhľadávania, napríklad v situáciách kedy sú na začiatku spojenia vedľa seba.

Literatúra

[1] J. F. DiMarzio: Practical Android 4 Games Development. Apress, 2011, ISBN: 978-1-4302-4029-7

[2] Reto Meier: Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275

[3] Sayed Hashimi: Pro Android 2, Apress, 2010, ISBN-13: 978-1430226598

[4] Android Developers [online]. [cit. 2015-04-19]. Dostupné z: http://developer.android.com/

[5] Yadvendra, Preeti: The video streaming over Wi-Fi network application client on the Android platform. [cit. 2015-04-19] Dostupné z: http://www.ijcsits.org/papers/vol3no42013/8vol3no4.pdf

[6] Darcey Lauren, Conder Shane: Android Wireless Application Development Volume 1: Android essentials, Addison-Wesley, 2012, ISBN 978-0-321-81383-1

[7] Darcey Lauren, Conder Shane: Android Wireless Application Development Volume 2: Advenced Topics, Addison-Wesley, 2012, ISBN-13: 978-0-321-81384-8

[8] Sessa Carlos: 50 Android Hacks, Manning Publications Co., 2013, ISBN 9781617290565

[9] Allen Grant: Beginning Android 4, Apress, 2012, ISBN-13: 978-1-4302-3985-7

[10] Austerberry David: The Technology of Video and Audio Streaming Second Edition, Taylor & Francis, 2005, ISBN 0240805801

[11] Donahoo J. Michael, Calvert L. Kenneth: TCP/IP Sockets in Java, Morgan Kaufmann, 29. 8. 2011, ISBN: 978-0-12-374255-1

[12] FoxDog Studios: Peepers: realtime video streaming from Android [online], dostupné z https://foxdogstudios.com/peepers

[13] Bovik Al: The Essential Guide to Video Processing, Academic Press, 7. 7. 2009, ISBN: 978-0-12-374456-2

[14] Barnes Ronald: Motion JPEG 47 Success Secrets, Emereo Publishing, 4. 11. 2014,

[15] Richardson E. Iain: Video Codec Design: Developing Image and Video Compression Systems, John Wiley & Sons, 17. 6. 2002, ISBN 0 471 48553 5

[16] Network Protocols Handbook, Javvin Technologies Inc., 1. 1. 2005, ISBN 408-872-3881

[17] WhatsApp Web [online]. [cit. 2015-04-02]. Dostupné z https://web.whatsapp.com

[18] Joe Follansbee: Streaming Media, Taylor & Francis, 2006, ISBN-10: 0-240-80863-0

[19] Wheeler Winston Dixon: Streaming movies, media and instant access, University Press of Kentucky, 19. 4. 2013, ISBN 978-0-8131-4217-3

[20] Viber [online]. [cit. 2015-04-02]. Dostupné z http://www.viber.com/en/

[21] Skype [online]. [cit. 2015-04-02]. Dostupné z http://www.skype.com/en/

[22] Line [online]. [cit. 2015-04-02]. Dostupné z http://line.me/en/

[23] Voxer [online]. [cit. 2015-04-02]. Dostupné z http://www.voxer.com

[24] Zello [online]. [cit. 2015-04-02]. Dostupné z http://zello.com

Zoznam obrázkov

Zoznam zdrojových kódov

Zoznam tabuliek